Computing a high-dimensional euclidean embedding from an arbitrary smooth riemannian metric

Zichun Zhong, Wenping Wang, Bruno Lévy, Jing Hua, Xiaohu Guo

Computing a High-Dimensional Euclidean Embedding from an Arbitrary Smooth Riemannian Metric

Zichun Zhong* Zichun Zhong* Xiaohu Guo§
Wenping Wang† Bruno Lévy‡ Jing Hua*
*Wayne State University §University of Texas at Dallas
†The University of Hong Kong ‡INRIA Nancy - Grand Est

Jan. 2018

Abstract

This article presents a new method to compute a *self-intersection free high-dimensional Euclidean embedding (SIFHDE)* for surfaces and volumes equipped with an arbitrary Riemannian metric. It is already known that given a high-dimensional (high-d) embedding, one can easily compute an anisotropic Voronoi diagram by back-mapping it to 3D space. We show here how to solve the inverse problem, i.e., given an input metric, compute a smooth intersection-free high-d embedding of the input such that the pullback metric of the embedding matches the input metric. Our numerical solution mechanism matches the deformation gradient of the 3D $\rightarrow$ higher-d mapping with the given Riemannian metric. We demonstrate applications of the method, by being used to construct anisotropic Restricted Voronoi Diagram (RVD) and anisotropic meshing, that are otherwise extremely difficult to compute. In the SIFHDE-space constructed by our algorithm, difficult 3D anisotropic computations are replaced with simple Euclidean computations, resulting in an isotropic RVD and its dual mesh on this high-d embedding. The results are compared with the state-of-the-art in anisotropic surface and volume meshings using several examples and evaluation metrics.

1 Introduction

Anisotropic meshes are important for not only improving the accuracy of the numerical simulations [AL10, NSO12] but also better approximating the shapes [Sim94].

*Zichun Zhong, Wenping Wang, Bruno Levy, Jing Hua, Xiaohu Guo *zichunzhong@wayne.edu, §xguo@utdallas.edu, †wenping@cs.hku.hk, ‡bruno.levy@inria.fr, *jinghua@wayne.edu

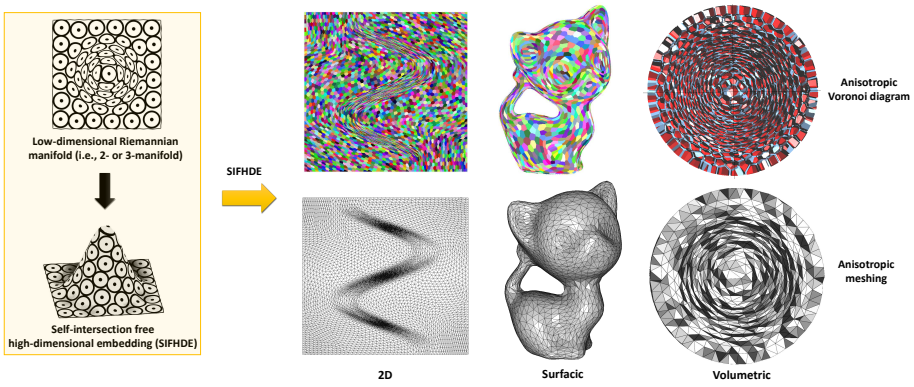

Figure 1: Illustration of self-intersection free high-dimensional embedding (SIFHDE) for anisotropic Restricted Voronoi Diagram and meshing on Riemannian 2- and 3-manifolds. Left: For a simple example of a 2D Riemannian manifold with a metric field, we can compute a SIFHDE of it in a 3D Euclidean space. The unit circles and bean-shaped curves are iso-distance contours in metric domains. Based on the proposed SIFHDE framework, a high-dimensional embedding for an arbitrary 2- or 3-manifold can be computed. Then, we can generate the Voronoi diagrams and meshes in such embedding with Euclidean metric. Finally, when they are mapped to the original domains, the anisotropic Voronoi diagrams (Top Right) and the anisotropic meshes (Bottom Right) will exhibit the desired anisotropy. Note that Right images show anisotropic Voronoi diagrams and meshes generated by our method in different scenarios, such as metrics defined by a 2D analytic function, 3D surface curvature, and 3D analytic function.

Unlike isotropic meshes, where the elements are chosen to be as regular and uniform as possible, anisotropic meshes are designed with elongated mesh elements that follow user specified orientations and aspect ratios.

Anisotropic meshes are notoriously difficult to compute. A possible strategy to overcome the difficulty is to map the (complicated) anisotropic 3D space onto a higher-d space, where geometric computations are made simpler (Euclidean / isotropic). In a certain sense, "anisotropy is traded for additional dimensions". This idea is inspired by the celebrated Nash embedding theorem, that states that every Riemannian manifold can be isometrically embedded into some high-dimensional (high-d) Euclidean space [Nas54, Kui55]. In such high-d embedding space, the metric is uniform and isotropic. When the isotropic mesh is computed in this embedding space and back-projected, the mesh elements on the original manifold will exhibit the desired anisotropic property. Recently, a few methods investigated this problem on surface meshing. Zhong et al. [ZGW+13] introduced a particle-based meshing approach that conceptualizes the inter-particle energy optimization in a high-d "embedding space". This embedding space was used in their energy and force formulation without computing such a space explicitly. However, their method has some limitations due to lack of explicit embedding. Specifically, without an explicit embedding, K-Nearest Neighbors (K-NN) search of particles and anisotropic Restricted Voronoi Diagram (RVD) computation under a Riemannian metric space are very challenging and time-consuming. Another group of research focused on computing explicit embeddings, such as 3D embedding [PPTSH14], 5D or 6D embedding [LB12, DMS14, DSPS15]. However, these methods have their own limitations, such as exhibiting self-intersections [PPTSH14], working only on specific dimensions [PPTSH14, LB12, DMS14, DSPS15], or using only specific embeddings (i.e., not taking into account arbitrary input Riemannian metrics) [LB12, DMS14, DSPS15].

The novelty introduced by our method is that our input is a smooth arbitrary user-defined metric, from which the embedding is deduced, whereas previous work observed that some predefined shapes of the embedding result in some specific anisotropies when back-mapped to 3D space. Our *main contribution* is to provide a numerical method that solves the "inverse problem". It makes our method much more general in terms of supported metrics. Besides that, previous work considered the surfacic case, whereas we also experiment with the (significantly more challenging) *volume* meshing problem. Our algorithm works with arbitrary embedding dimension and avoids self-intersections. On the computed embedded surface or volume, we can directly optimize for a uniform, isotropic particle distribution and generate a high-quality isotropic RVD and its dual mesh on it. When mapped back to the original surface or volume, an anisotropic RVD as well as a simplicial mesh are obtained. Fig. 1 illustrates the method in the flat 2D setting, 3D surface setting and volumetric setting.

The embedding problem is formulated as an optimization that minimizes the deviation between the given metric and the *deformation gradient* of a map from the original surface / volume to the high-d embedded surface / volume. Our empirical experiments detailed further demonstrate that our method constructs

an embedding with a small metric deviation error. The benefits of the high-d embedding are as follows:

- the so-constructed high-d space can be used to compute anisotropic Voronoi diagrams for surfaces and volumes with a prescribed anisotropy. To our knowledge, this is the first time that an effective computational algorithm is proposed to construct such objects in the volumetric case with arbitrary metric;
- it introduces new computational strategies for some recent anisotropic meshing algorithms, such as particle optimization, RVD, and dual meshing in the high-d Euclidean embedding space. Our empirical evaluations in several scenarios demonstrate a practical utility, using a 8D embedding space.

2 Related Work

2.1 Anisotropic Triangulation

Anisotropic Centroidal Voronoi Tessellation: In order to compute well sampled anisotropic meshes, Du et al. [DW05a] further generalized the concept of Centroidal Voronoi Tessellation (CVT) to the anisotropic case – Anisotropic CVT (ACVT), and then computed its dual mesh graph. An Anisotropic Voronoi Diagram (AVD) with the given Riemannian metric needs to be constructed in each Lloyd iteration [Llo82], which is a time-consuming operation. To make the computation faster, Valette et al. [VCP08] described a discrete approximation of ACVT by clustering the vertices of a dense pre-triangulation of the domain, at the expense of degraded mesh quality. Recently, Zhong et al. [ZSJG14] provided a method to solve the anisotropic meshing by conformally mapping the metric surface to an appropriate 2D parametric domain and then compute CVT on it. But the limitation of the conformal embedding method is that it cannot handle surfaces with complicated topologies.

Surface Meshing in Higher Dimensional Space: Several solutions were proposed for certain classes of surface meshing problems by embedding them in high-d spaces [CnG06, BWY08a, KMZ10, LB12]. Lévy and Bonneel [LB12] extended the computation of CVT to a 6D space in order to achieve the curvature-adaptation. The main idea of their method is to transform the anisotropic meshing problem on a 3D surface to an isotropic one embedded in 6D space described by vertex positions and normals. Recently, Dassi et al. [DMS14, DSPS15] used Lévy and Bonneel's framework to compute remeshing, but instead of optimizing the CVT in 6D, they used local operations (i.e., edge flips). It should be noted that both Lévy and Bonneel's and Dassi et al.'s work does not provide user's flexibility to control the anisotropy through an input arbitrary metric field.

Particle-Based Anisotropic Meshing: Particle-based approaches for anisotropic meshing have been proposed during the past two decades. Bossen and Heckbert [BH96] simulated the repulsion and attraction forces between particles based on a distance function with the metric tensor. Shimada et

al. [SG95, SYI97] physically modeled the inter-bubble forces by a bounded cubic function of the distance, and further extended it to anisotropic meshing by converting spherical bubbles to ellipsoidal ones. Both Bossen et al. and Shimada et al.'s work requires dynamic population control schemes, that is to adaptively insert or delete particles / bubbles in certain regions. Zhong et al. [ZGW$^+$13] showed that by formulating the inter-particle energy optimization in a high-d space, such optimizations have very fast convergence without any need for the explicit control of particle population. But they still needed to compute the anisotropic RVD on the surface in 3D space in their final step of mesh generation, which is less robust and efficient. Another bottleneck in their framework is the search speed of neighboring particles. When the anisotropic stretching ratio is high, it uses a large set of K-NN candidates in the original space that contains many false positives. A comparison with our approach is shown in Sec. 6.

Refinement-Based Anisotropic Delaunay Triangulation: Delaunay refinement-based approaches involve iterative point insertion in a Delaunay triangulation. Many researchers have extended these approaches to work with anisotropic triangulations, and there are some practical applications [BGH$^+$97, BGM97, DF08]. Cheng et al. [CDES01, CDRR04, CDR06] applied Delaunay refinement to anisotropic surface meshing. Boissonnat et al. [BWY08b, BWY15, BSTY14] introduced a Delaunay refinement framework, which makes the star around each vertex $\mathbf{x}_i$ to be consisting of the triangles that are exactly Delaunay for the metric associated with $\mathbf{x}_i$. In order to "stitch" the stars of neighboring vertices, refinement algorithms add new vertices gradually to reach the final anisotropic meshes. Recently, Rouxel-Labbé et al. [RLWB16] introduced an algorithm to compute discrete approximations of Riemannian Voronoi diagrams on 2-manifolds by using the numerical computation of geodesic distances. The main strength of these methods is that they come with theoretical proofs, both on termination (finite number of inserted points) and existence of a straight dual. However, since they use *local* criteria to insert points, the final result is often of lower quality than when using a *global* optimization [ZGW$^+$13, FLSG14]. Since they use the same representation of the metric as in [LB12] and as us (piecewise constant in the simplices), their method can be used as a *post-processing* after a global optimization, in order to benefit from both advantages (quality of global optimization and robustness / theoretical guarantees of Delaunay refinement).

Optimal Delaunay Triangulation: Another idea for anisotropic meshing is through Delaunay-based variational approach, by minimizing the error between a lifted quadratic target function and the linear interpolation of the constructed mesh. Chen et al. [CX04, CSX07] proposed Optimal Delaunay Triangulation (ODT) method to minimize the error function for both isotropic and anisotropic meshing. Alliez et al. [ACSYD05] presented a robust approach for isotropic tetrahedral meshing by exploiting the ODT formalism. In the context of varying densities, the objective function was carefully analyzed and improved in [CWL$^+$14]. ODT can be considered as lifting the points onto a paraboloid and estimating volume integrals on this paraboloid. Budninskiy et al. [BLdG$^+$16] presented a variational method to take into account an anisotropy by using a

convex function instead of a paraboloid. While elegant and interesting from a theoretical point of view, their method is limited to a small class of anisotropies stemming from convex functions. The constraint is not reasonable in practical applications (for instance, it could not account for the anisotropy in Fig. 1-Right, nor with any non-convex geometry). Fu et al. [FLSG14] proposed a Locally Convex Triangulation (LCT) method to compute the anisotropic meshes based on constructing convex functions that locally match the specified Riemannian metric. We compare the quality of our generated mesh with Fu et al.'s method in Sec. 6.

2.2 Anisotropic Tetrahedralization

Theoretically, it is possible to extend ACVT-based method for anisotropic meshing from 2D domain or surface [DW05a, VCP08, ZSJG14] into 3D volume, but this group of methods needs to compute AVD iteratively in the ambient 3D space with specified metrics, which makes the computation very complicated and impractical. In computer graphics and geometric modeling areas, there is few literature in anisotropic tetrahedral meshing: Dobrzynski and Frey's local mesh adaptation method based on an extension of Delaunay kernel [DF08], Klingner et al.'s aggressive optimization [KS07, Kli08] and Fu et al.'s LCT method [FLSG14] on some simple 3D models. Yamakawa and Shimada [YS00] proposed an ellipsoidal bubble packing method, but inserting or deleting particles / bubbles is necessary in the computation. In mechanical and biomedical engineering, anisotropic meshing is widely used in computational fluid dynamics (CFD) [FA05, AL16, LL16] and blood flow simulations [MGR13, SRM14]. However, their methods are numerical and error estimation-based approaches. For instance, they are very time-consuming, since partial differential equations (PDEs) are solved at each solver time step, which make it difficult to efficiently compute complicated 3D anisotropic volume meshes.

2.3 Other Related Embedding Work

Bronstein et al. [BBKY00, BBKY05] demonstrated the performance of multi-dimensional scaling (MDS). Instead of raising the dimensions, they mapped the original metric domain into a parameter domain (similar to a conformal embedding). Verma [Ver12] proposed the distance preserving embeddings for general n-dimensional manifolds in the machine learning field. It applies "spiraling perturbations / corrections" for globally isometric mapping. This cannot compute a smooth high-d embedding for our computer graphics research and applications. Recently, Panozzo et al. [PPTSH14] proposed a 3D embedding framework with self-intersections to compute a surface deformation that warps a frame field into a cross field, and used in the adaptive quad meshing. In their method, the parameterization is used to handle the final meshing with self-intersections. However, our proposed method does not need an extra step to parameterize the embedding for meshing, since we can directly produce the Voronoi diagrams and meshes on the computed SIFHDE. Another limitation of

their method is that the frame-driven deformation for 3D embedding computation may deteriorate the quality of the input triangulation, especially when the stretching ratio is high, such as ≥ 10, which leads to failure. However, our method theoretically can compute the embedding for any high-stretching ratio case, since we provide enough degrees of freedom in higher dimensions. We compare the embedding quality of our method with Panozzo et al.'s method in Sec. 6. Besides surface cases, our proposed high-d embedding approach can work on volume cases, which is an unexplored topic in the previous work.

3 High-D Embedding

3.1 Anisotropy and High-D Embedding

We consider that a metric $\mathcal{M}(.)$, i.e., a symmetric positive definite (SPD) bilinear form, is defined over the surface or volume domain $\Omega \subset \mathbb{R}^m$. In other words, at a given point $\mathbf{x} \in \Omega$, the *dot product* between two vectors $\mathbf{a}$ and $\mathbf{b}$ is given by $\langle \mathbf{a}, \mathbf{b} \rangle_{\mathcal{M}(\mathbf{x})}$. The metric can be represented by a symmetric $m \times m$ matrix $\mathbf{M}(\mathbf{x})$, in which case the dot product becomes:

$$\langle \mathbf{a}, \mathbf{b} \rangle_{\mathcal{M}(\mathbf{x})} = \mathbf{a}^T \mathbf{M}(\mathbf{x}) \mathbf{b}. \tag{1}$$

Through Singular Value Decomposition (SVD), the metric matrix $\mathbf{M}(\mathbf{x})$ can be decomposed into:

$$\mathbf{M}(\mathbf{x}) = \mathbf{R}(\mathbf{x})^T \mathbf{S}(\mathbf{x})^2 \mathbf{R}(\mathbf{x}), \tag{2}$$

where the diagonal matrix $\mathbf{S}(\mathbf{x})^2$ contains its ordered eigenvalues (i.e., a scaling field), and the orthogonal matrix $\mathbf{R}(\mathbf{x})$ contains its eigenvectors (i.e., a rotation field). In the surface case [DW05a, ZGW+13], they are defined on the tangent spaces of the surface. $\frac{s_2}{s_1}$ is defined as *stretching ratio* in the surface metric field $\mathbf{M}(\mathbf{x})$, where s_1 and s_2 are the two diagonal items in $\mathbf{S}(\mathbf{x})$ and $s_1 \leq s_2$. Similarly, in volume case [YS00], they represent the scaling and rotation in a 3D volume space.

Metric through High-D Embedding: For an arbitrary metric field $\mathbf{M}(\mathbf{x})$ defined on the surface or volume $\Omega \subset \mathbb{R}^m$ (i.e., Riemannian 2- or 3-manifold), Nash theorem [Nas54] states that there exists a high-d space $\mathbb{R}^{\overline{m}}$ (i.e., $m \leq \overline{m}$), in which the surface or volume can be embedded with Euclidean metric as $\overline{\Omega} \subset \mathbb{R}^{\overline{m}}$. In this article, $\overline{\Omega}$ denotes the "embedded surface or volume". Consider a mapping denoted by $\phi : \Omega \to \overline{\Omega}$.

From the definition of anisotropy above, it can be seen that introducing anisotropy means changing the definition of the dot product. If we consider two vectors $\mathbf{a}$ and $\mathbf{b}$ from a given location $\mathbf{x} \in \Omega$, then they are transformed into $\overline{\mathbf{a}} = \mathbf{J}(\mathbf{x})\mathbf{a}$ and $\overline{\mathbf{b}} = \mathbf{J}(\mathbf{x})\mathbf{b}$ on the high-d embedded surface or volume $\overline{\Omega}$, where $\mathbf{J}(\mathbf{x})$ denotes the Jacobian matrix of mapping ϕ at $\mathbf{x}$. The dot product between $\overline{\mathbf{a}}$ and $\overline{\mathbf{b}}$ is given by the *pullback metric* of Φ, defined as:

$$\langle \overline{\mathbf{a}}, \overline{\mathbf{b}} \rangle = \mathbf{a}^T \mathbf{J}(\mathbf{x})^T \mathbf{J}(\mathbf{x}) \mathbf{b} = \mathbf{a}^T \mathbf{M}(\mathbf{x}) \mathbf{b}. \tag{3}$$

Importance of High-D Embedding: (1) In a high-d space, more degrees of freedom are available to deform and embed the given surface or volume, so that better embedding quality is obtained. Experiments are shown in Sec. 6. (2) When using the high-d space, we can avoid self-intersections of the embedded surface or volume as discussed in Sec. 3.4.2, instead of embedding them in the original space. (3) By generating a high-d Euclidean embedding without self-intersections, we are able to drastically simplify several Riemannian geometric applications, such as computing anisotropic RVD and meshing on surface and volume with high-quality elements using only isotropic Euclidean computations. This is both more efficient and more accurate than the computations in metric space. Details are presented in Sec. 4.

3.2 High-D Embedding Transformation

In this work, we assume the surfaces are all represented as triangle meshes and the volumes are all represented as tetrahedron meshes. The two surfaces / volumes Ω (in the original space) and $\overline{\Omega}$ (in the high-d space) share the same number of vertices and the same connectivity between vertices. The only difference between them is their vertex coordinates and dimensions; thus, we assume that the two surfaces / volumes have the same indices of vertices and triangles / tetrahedrons. In the following subsection, we discuss the surface and volume cases, respectively.

Surface: For a triangle j on Ω and $\overline{\Omega}$, let $\{\mathbf{v}_{j_1},\mathbf{v}_{j_2},\ \mathbf{v}_{j_3}\}$ and $\{\overline{\mathbf{v}}_{j_1},\ \overline{\mathbf{v}}_{j_2},\ \overline{\mathbf{v}}_{j_3}\}$ denote their vertices, respectively. Their corresponding edge vectors can be represented as the following two matrices $\mathbf{W}_j$ and $\overline{\mathbf{W}}_j$:

$$\begin{aligned} \mathbf{W}_j &= [\mathbf{v}_{j_2} - \mathbf{v}_{j_1}, \mathbf{v}_{j_3} - \mathbf{v}_{j_1}] \\ \overline{\mathbf{W}}_j &= [\overline{\mathbf{v}}_{j_2} - \overline{\mathbf{v}}_{j_1}, \overline{\mathbf{v}}_{j_3} - \overline{\mathbf{v}}_{j_1}]. \end{aligned} \tag{4}$$

Volume: For a tetrahedron j on Ω and $\overline{\Omega}$, let $\{\mathbf{v}_{j_1},\ \mathbf{v}_{j_2},\ \mathbf{v}_{j_3},\ \mathbf{v}_{j_4}\}$ and $\{\overline{\mathbf{v}}_{j_1},\ \overline{\mathbf{v}}_{j_2},\ \overline{\mathbf{v}}_{j_3},\ \overline{\mathbf{v}}_{j_4}\}$ denote their vertices, respectively. Their corresponding edge vectors can be represented as the following two matrices $\mathbf{W}_j$ and $\overline{\mathbf{W}}_j$:

$$\begin{aligned} \mathbf{W}_j &= [\mathbf{v}_{j_2} - \mathbf{v}_{j_1}, \mathbf{v}_{j_3} - \mathbf{v}_{j_1}, \mathbf{v}_{j_4} - \mathbf{v}_{j_1}] \\ \overline{\mathbf{W}}_j &= [\overline{\mathbf{v}}_{j_2} - \overline{\mathbf{v}}_{j_1}, \overline{\mathbf{v}}_{j_3} - \overline{\mathbf{v}}_{j_1}, \overline{\mathbf{v}}_{j_4} - \overline{\mathbf{v}}_{j_1}]. \end{aligned} \tag{5}$$

Their relationship can be represented as:

$$\overline{\mathbf{W}}_j = \mathbf{J}_j \mathbf{W}_j, \tag{6}$$

where $\mathbf{J}_j$ is the Jacobian transformation matrix for triangle or tetrahedron j.

In what follows, we assume that the original metric is smooth over the surface or the volume, and sampled as a constant symmetric matrix $\mathbf{M}_j$ attached to each mesh simplex j. Note that in practice, it may be specified in the input at each vertex. In this case, several techniques can be used to interpolate the metric and deduce a reasonable constant value on each simplex (see [CLGD06] for a discussion).

We now elaborate on the relationship between the Jacobian matrix of the mapping $\mathbf{J}_j$ and the metric $\mathbf{M}_j$.

To better understand $\mathbf{J}_j$, it is necessary to explore the relationship between the original triangle or tetrahedron j on Ω and the embedded triangle or tetrahedron j on $\overline{\Omega}$. $\mathbf{J}_j$ is an $\overline{m} \times m$ matrix, and it can be represented as the product of a rotation in the high-d embedding space, and a scaling and rotation in the original space. That is:

$$\overline{\mathbf{W}}_j = \overline{\mathbf{U}}_j \begin{bmatrix} \mathbf{S}_j \mathbf{R}_j \\ \mathbf{0} \end{bmatrix} \mathbf{W}_j, \tag{7}$$

where $\overline{\mathbf{U}}_j$ is an $\overline{m} \times \overline{m}$ unitary matrix (i.e., a rotation matrix in $\mathbb{R}^{\overline{m}}$); $\mathbf{0}$ is a $(\overline{m} - m) \times m$ block of zeros; $\mathbf{S}_j$ and $\mathbf{R}_j$ are the diagonal scaling matrix and rotation matrix extracted from SVD of the metric $\mathbf{M}_j$ as shown in Eq. (2).

Thus, the embedding transformation $\mathbf{J}_j$ is:

$$\mathbf{J}_j = \overline{\mathbf{U}}_j \begin{bmatrix} \mathbf{S}_j \mathbf{R}_j \\ \mathbf{0} \end{bmatrix}. \tag{8}$$

By denoting $\mathbf{Q}_j = \begin{bmatrix} \mathbf{S}_j \mathbf{R}_j \\ \mathbf{0} \end{bmatrix}$, we can simply represent $\mathbf{J}_j$ as:

$$\mathbf{J}_j = \overline{\mathbf{U}}_j \mathbf{Q}_j. \tag{9}$$

3.3 High-D Deformation Gradient

Intuitively, the transformation between the original surface or volume Ω and its high-d embedded one $\overline{\Omega}$ can be considered as the deformation of Ω. It is represented by the field of the *deformation gradient* over the surface or volume as being intuitively by Sumner and Popović [SP04]. This field is defined per triangle face or tetrahedron volume. We extend their idea from 3D surface to our high-d embedding as well as volume cases. Given a manifold with no deformation (i.e., the original surface or volume Ω) and its corresponding deformed manifold (i.e., the embedded surface or volume $\overline{\Omega}$), the deformation gradient $\mathbf{T}_j$ for triangle or tetrahedron j ($\mathbf{T}_j$ is an $\overline{m} \times m$ matrix) can be defined by:

$$\mathbf{T}_j \mathbf{W}_j = \overline{\mathbf{W}}_j. \tag{10}$$

In order to obtain $\mathbf{T}_j$, we need to compute the inverse of $\mathbf{W}_j$. Since the surface and volume have different representations of edge vectors, there are two cases as follows:

(1) Surface: As for the surface, $\mathbf{W}_j$ is a 3×2 matrix, so we cannot directly compute its inverse matrix. With QR factorization of $\mathbf{W}_j$ [GL96]:

$$\mathbf{W}_j = \mathbf{O}_j \begin{bmatrix} \mathbf{P}_j \\ \mathbf{0} \end{bmatrix} = \begin{bmatrix} \mathbf{O}_{j\alpha} \mathbf{O}_{j\beta} \end{bmatrix} \begin{bmatrix} \mathbf{P}_j \\ \mathbf{0} \end{bmatrix} = \mathbf{O}_{j\alpha} \mathbf{P}_j, \tag{11}$$

where $\mathbf{P}_j$ is a 2×2 upper triangular matrix, $\mathbf{O}_j$ is an $m \times m$ orthogonal matrix, $\mathbf{O}_{j\alpha}$ is an $m \times 2$ matrix, and $\mathbf{O}_{j\beta}$ is an $m \times (m-2)$ matrix, where $m = 3$. Then we have the minimum norm solution of the deformation gradient for triangle j:

$$\mathbf{T}_j = \overline{\mathbf{W}}_j \mathbf{P}_j^{-1} \mathbf{O}_{j\alpha}^T. \tag{12}$$

(2) Volume: As for the volume, $\mathbf{W}_j$ is a 3×3 square matrix, so the inverse of it can be directly computed as:

$$\mathbf{T}_j = \overline{\mathbf{W}}_j \mathbf{W}_j^{-1}. \tag{13}$$

3.4 Embedding Optimization

From Eq. (9) and Eq. (12) (or Eq. (13)), we can clearly see that in essence, the high-d embedding transformation $\mathbf{J}_j$ and the high-d deformation gradient $\mathbf{T}_j$ are the same. For $\mathbf{J}_j$ in Eq. (9), $\mathbf{Q}_j$ is coming from the given metric $\mathbf{M}_j$ on the surface or volume, while $\overline{\overline{\mathbf{U}}}_j$ is the high-d rotation matrix that is unknown. For $\mathbf{T}_j$ in Eq. (12), $\mathbf{P}_j$ and $\mathbf{O}_{j\alpha}$ are from the edge vector $\mathbf{W}_j$ of the original surface, or in Eq. (13), $\mathbf{W}_j^{-1}$ is from the edge vector $\mathbf{W}_j$ of the original volume, while $\overline{\overline{\mathbf{W}}}_j$ is the edge vector of the high-d embedded triangle or tetrahedron that is unknown.

Thus, knowing either of these two matrices, i.e., $\mathbf{J}_j$ and $\mathbf{T}_j$, guides us in computing the other. We can formulate an expression to minimize the function E_{em} defined as the difference between these two matrices:

$$E_{em}(\overline{\mathbf{v}}_1, \cdots, \overline{\mathbf{v}}_{n_v}) = \min \sum_{j=1}^{n_{ele}} \|\mathbf{T}_j - \mathbf{J}_j\|_F^2, \tag{14}$$

where n_v is the number of vertices, n_{ele} is the number of mesh elements, i.e., triangles or tetrahedrons, and $\{\overline{\mathbf{v}}_1, \cdots, \overline{\mathbf{v}}_{n_v}\}$ are the vertices of the high-d embedded surface or volume $\overline{\Omega}$. The matrix norm $||.||_F$ is the Frobenius norm.

By substituting Eq. (9) and Eq. (12) (or Eq. (13)) into Eq. (14), we can get the more detailed expressions for this objective function :

Surface:

$$E_{em}(\overline{\mathbf{v}}_1, \cdots, \overline{\mathbf{v}}_{n_v}) = \min \sum_{j=1}^{n_{ele}} \left\|\overline{\overline{\mathbf{W}}}_j \mathbf{P}_j^{-1} \mathbf{O}_{j\alpha}^T - \overline{\overline{\mathbf{U}}}_j \mathbf{Q}_j\right\|_F^2. \tag{15}$$

Volume:

$$E_{em}(\overline{\mathbf{v}}_1, \cdots, \overline{\mathbf{v}}_{n_v}) = \min \sum_{j=1}^{n_{ele}} \left\|\overline{\overline{\mathbf{W}}}_j \mathbf{W}_j^{-1} - \overline{\overline{\mathbf{U}}}_j \mathbf{Q}_j\right\|_F^2. \tag{16}$$

3.4.1 Regularization Term

It is well known that solutions of Nash equations often present wrinkles, known as corrugations, that can form fractal patterns (see [BJLT12] and references herein on convex integration). Thus, a direct minimization of the criterion in Eq. (15) or Eq. (16) may lead to an oscillating high-d embedding, with wrinkles, that may have an influence on the stability / robustness of the subsequent geometric operations: in the high-d embedding space, we would like to use the Euclidean distance to approximate the metric distance on the original surface or volume, in order to efficiently compute: (1) K-NN for particle optimization in the high-d space; and (2) high-d RVD and dual mesh based on uniformly optimized particles. Thus the embedded surface or volume has to be smooth enough, in order to ensure that the K-NN and RVD computed with Euclidean distances in the high-d space are sufficiently accurate.

Thus, to regularize the embedding, we add the following term:

$$E_{reg}(\overline{\mathbf{v}}_1, \cdots, \overline{\mathbf{v}}_{n_v}) = \sum_{i=1}^{n_{v-b}} \sum_{d=1}^{\overline{m}} (\frac{\sum_{k \in N(i)} (\overline{v}_k^d - \overline{v}_i^d)}{|N(i)|})^2, \tag{17}$$

where n_{v-b} is the total number of vertices excluding those on the open boundaries in surface case or boundary surface in volume case, $N(i)$ is the set of one-ring neighbors of vertex i, $|N(i)|$ is the size of set $N(i)$, and $\overline{v}_i^d$, $\overline{v}_k^d$ are the d-th dimensional coordinates of vertices $\overline{\mathbf{v}}_i$ and $\overline{\mathbf{v}}_k$. The regularity term E_{reg} is a summation of the square of Graph Laplacian operations over every vertex in the embedding space except those vertices on the open boundaries in surface case or boundary surface in volume case.

Now our regularized objective function for the embedding optimization includes two terms: the similarity between two transformations and the regularity used to achieve smoothness of the embedding:

$$E_{total} = E_{em} + \mu E_{reg}, \tag{18}$$

where μ is a weighting factor to balance the similarity and regularity terms during optimization. In our experiments, μ is set to be 100.

3.4.2 Avoiding Intersections and Choosing Target Dimension

According to Nash embedding theorem, using the mapping $\phi : \Omega \to \overline{\Omega}$ given by $\mathbf{v}^{1:m} \to (\mathbf{v}^{1:m}, \overline{\mathbf{v}}^{m+1}, \cdots, \overline{\mathbf{v}}^{\overline{m}})$, one obtains an embedding without self-intersections. In our case, to avoid self-intersections that may come from numerical approximations or from the regularization term, in addition we keep the original 3D coordinates, thus, in a certain sense, the additional nD coordinates that we compute act as "correcting terms" to solve for the metric. Clearly, if the original 3D surface is free of self-intersections, then it is also the case of our (n+3)D embedding. Compared with methods that avoid self-intersections a-posteriori in a 3D embedding [PPTSH14], this is both simpler, and leaves more degrees of freedom to fit the user-specified metric. Generally speaking, the higher the dimension is, the smaller the embedding error is, since more degrees of freedom are provided to deform and embed the given surface or volume in such an embedding space; it is necessary to choose the dimension in a way that balances the computational efficiency and the embedding errors. Our empirical results showed that 8D is a reasonable target dimension for embedding most general metric surfaces and volumes in small errors and without any self-intersection. More details and empirical validation will be given in Sec. 6 (particularly in Sec. 6.1.2).

3.4.3 Numerical Solution Mechanism

The optimization defined above is a non-linear problem with two unknown parameters (i.e., $\overline{\mathbf{W}}_j$ and $\overline{\mathbf{U}}_j$). We use an iterative method to obtain the optimal solution of $\overline{\mathbf{W}}_j$. It should be noted that the optimized vertex coordinates

$\overline{\mathbf{v}}_1, \cdots, \overline{\mathbf{v}}_{n_v}$ of the embedded surface or volume are included in the edge vectors $\overline{\mathbf{W}}_j$.

Initially, for each vertex i, we set its high-d embedding coordinates as $\overline{\mathbf{v}}_i = (\mathbf{v}_i^{1:m}, \overline{v}_i^{m+1}, \cdots, \overline{v}_i^{\overline{m}})$, where the first m dimensions are the same as the original surface or volume's coordinates, and use a small random perturbation to each higher dimensional coordinate (from dimension $m+1$ to dimension $\overline{m}$), i.e., not all equal to zero. The insightful reason is that at first we fix $\overline{\mathbf{W}}_j$ to compute the rotation matrix $\overline{\mathbf{U}}_j$ in high-d space. $\overline{\mathbf{U}}_j$ is computed from Eq. (19) or Eq. (20) as follows, which includes $\overline{\mathbf{W}}_j$. If the higher dimensional coordinates are all zeros, $\overline{\mathbf{W}}_j$ will be degraded in the original space $\mathbb{R}^m$, leading to the high-d rotation $\overline{\mathbf{U}}_j$ restricted in the original space as well, so that we use this trick to add the small perturbation value to let the optimization "jump" out of the original space.

When the vertex coordinates $\{\overline{\mathbf{v}}_i | i = 1, \cdots, n_v\}$ are fixed (i.e., all $\overline{\mathbf{W}}_j$s are fixed), the optimal $\overline{\mathbf{U}}_j$ can be computed as the orthogonal factor of the following Polar decomposition:

Surface:

$$\overline{\mathbf{U}}_j = Polar(\overline{\mathbf{W}}_j \mathbf{P}_j^{-1} \mathbf{O}_{j\alpha}^T \mathbf{Q}_j^T). \tag{19}$$

Volume:

$$\overline{\mathbf{U}}_j = Polar(\overline{\mathbf{W}}_j \mathbf{W}_j^{-1} \mathbf{Q}_j^T). \tag{20}$$

The detailed derivations are given in the supplementary material (Appendix A).

When all $\overline{\mathbf{U}}_j$s are fixed, we can compute the optimal vertex coordinates $\{\overline{\mathbf{v}}_i | i = 1, \cdots, n_v\}$, where $\overline{\mathbf{v}}_i^{1:3} = \mathbf{v}_i^{1:3}$, by solving a linear system, since the objective function E_{total} is a quadratic form of the vertex coordinates. This is similar to the method used by Sumner and Popović [SP04]. Deformation gradients are invariant to translation, so there are infinite solutions to this optimization problem: all translations of one optimal solution are also optimal. Thus we fix the coordinates of the first vertex in the mesh in order to make the solution unique.

Our goal is to ensure that the regularized objective function always decreases during optimization. The optimization strategy is similar to [SA07, PPTSH14]. In summary, as shown in the following Alg. 1, we can optimize the vertex coordinates $\overline{\mathbf{v}}_1, \cdots, \overline{\mathbf{v}}_{n_v}$ of the embedded surface or volume iteratively, until convergence by satisfying a specified stopping condition, e.g., the magnitude of the gradient is smaller than a threshold or the desired number of iterations is reached. In our experiments, we use the number of iterations as 50 for the stopping condition. One example of the evolution of the objective function during optimization is given in Fig. 2 (b), which can reach a satisfactory small embedding error.

4 Anisotropic RVD and Meshing

In order to demonstrate the importance and usefulness of the proposed SIFHDE, we demonstrate two applications, i.e., computing anisotropic RVD and anisotropic

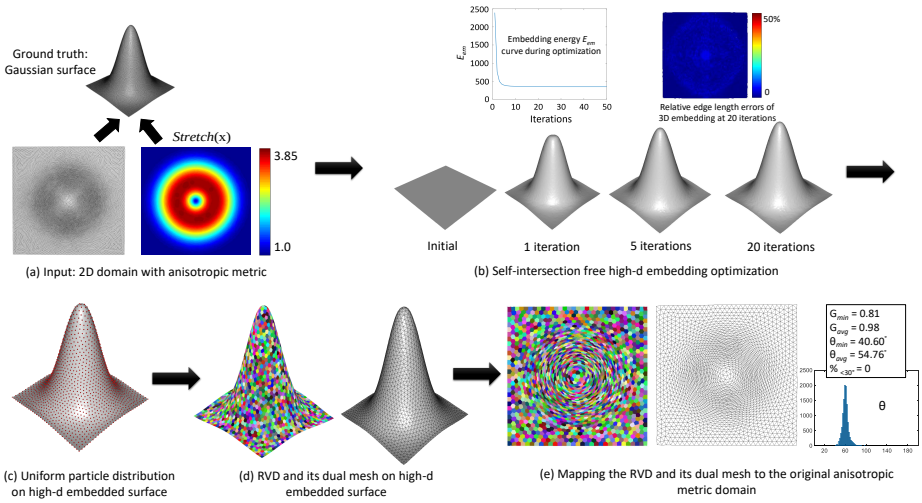

Figure 2: Illustration of the proposed high-d embedding for anisotropic RVD and surface meshing. (a) A simple example of a 2D domain equipped with anisotropic metric, which has an ideal 3D embedding – the Gaussian surface as the ground truth. (b) Our embedding optimization iteratively deforms the original domain into the embedded surface. After embedding we can (c) uniformly and isotropically sample particles and (d) compute the isotropic RVD and its dual mesh on the embedded surface. (e) When they are mapped to the original 2D domain, the RVD and the triangular mesh will exhibit the desired anisotropy.

triangular / tetrahedral meshing.

Since the high-d Euclidean embedding without self-intersections has been computed in the previous section (Sec. 3), in the following section, under the isotropic metric space, we use the particle optimization technique (Sec. 4.1) to compute isotropic RVD and its dual Delaunay triangulation on the high-d embedded surface or volume (Sec. 4.2), due to its efficiency to generate regular hexagonal patterns of particles on 2-manifold and regular body-centered-cubic (BCC) lattices on 3-manifold, which are similar to the results of CVT [DFG99, LWL+09, DW05b]. Moreover, the particle-based method is easily formulated in the high-d space. Finally the generated RVD and triangulation / tetrahedralization can be mapped from the high-d embedding space to the original space, on which the RVD and triangulation / tetrahedralization will exhibit the desired anisotropic property.

4.1 Particle Optimization with Efficient K-NN in High-D Embedding

In this section, we discuss the isotropic particle optimization on the explicitly-computed high-d embedded surface or volume, which is the main difference

Data: Original surface or volume Ω with user-specified metric $\mathbf{M}$ and target dimension $\overline{m}$
Result: Vertex coordinates $\{\overline{\mathbf{v}}_i | i = 1, \cdots, n_v\}$ of the embedded surface or volume $\overline{\Omega}$ in $\overline{m}$ space
Initialize vertex coordinates in $\overline{m}$ space;
while *stopping condition not satisfied* **do**
  for *each triangle or tetrahedron* j **do**
    Compute $\overline{\mathbf{U}}_j$ by fixing $\{\overline{\mathbf{v}}_i | i = 1, \cdots, n_v\}$ and using Eq. (19) or Eq. (20);
  end
  Compute $\{\overline{\mathbf{v}}_i | i = 1, \cdots, n_v\}$ by fixing all $\overline{\mathbf{U}}_j$s and fixing $\overline{\mathbf{v}}_1$, i.e., solve a linear system to minimize the regularized objective function E_{total} of Eq. (18);
end

Algorithm 1: High-D Embedding Optimization Framework.

between ours and Zhong et al.'s method [ZGW+13]. The limitations of their methods are: (1) they made use of the concept of embedding space for their energy and force formulations, without computing such high-d embedding explicitly, which may lead to more approximation error; (2) during the particle optimization, they had to have a very large range for searching neighboring particles in the original metric space when the anisotropic stretching ratio is high, which is less efficient. Besides that, they only worked on surface case without considering volume case.

However, based on our method, once the high-d embedding is computed as discussed in Sec. 3, we can directly define the inter-particle energy and force in such embedding space. The goal is to let particles reach isotropic uniform distribution at their equilibrium state, where the particle distribution on the original surface or volume will exhibit the desired anisotropic property.

In our method, we formulate the isotropic inter-particle energy and force with explicit coordinates of the particles in the high-d embedding space, which is based on [ZGW+13]. The energy $\overline{E}^{pq}$ between particles p and q in the high-d embedding space is defined as: $\overline{E}^{pq} = e^{-\frac{\|\overline{\mathbf{x}}_p - \overline{\mathbf{x}}_q\|^2}{4\sigma^2}}$, where σ is kernel width, and $\overline{\mathbf{x}}_p$ and $\overline{\mathbf{x}}_q$ are particle positions in the embedding space. Note that our goal is to let the particles uniformly and isotropically sampled on $\overline{\Omega}$, so $\sigma = 0.3\sqrt[d]{|\overline{\Omega}|/n}$, as suggested in [ZGW+13], where $d = 2$ in surface and $d = 3$ in volume, $|\overline{\Omega}|$ denotes the area or the volume of $\overline{\Omega}$ in the embedding space, n is the number of particles. The force applied from particle p to particle q in the embedding space is defined as the gradient of the inter-particle energy. To sum up all inter-particle energies, the computational complexity of particle optimization is $O(cn_p)$, where c is a constant, i.e., the average number of K-NN for each particle, and n_p is total number of particles (i.e., the number of RVD cells / final mesh vertices). The k-d tree data structure is used to compute K-NN. Since n_p is provided by user, c

is the key factor to affect the efficiency of the computation. With the computed high-d Euclidean embedded surface or volume, the particle positions are defined and optimized on such embedding, which helps us to determine the c around 20∼30, i.e., nearest neighbors in Euclidean metric space within five standard deviations of the Gaussian energy kernel (5σ). If without the high-d Euclidean embedding, hundreds or thousands of neighbors need to be checked and then prune the spurious neighbors when the stretching ratio in the given metric field is high, such as ≥ 10. The K-NN comparison experiments are given in Sec. 6.2. Finally, we use the L-BFGS algorithm [LN89] to optimize the particle positions constrained on the high-d embedded surface or inside the high-d embedded volume.

4.2 Restricted Voronoi Diagram and Mesh Generation

High-D RVD and Meshing: The input of high-d RVD computation are the triangle / tetrahedral mesh of the high-d embedded surface / volume $\overline{\Omega}$ and the sites (i.e., optimized particle positions in high-d space) of high-d Voronoi cells. The key task is to identify the high-d Voronoi cells that overlap each triangle / tetrahedron of the embedded surface / volume $\overline{\Omega}$ and compute their intersections. The RVD computation on surface is based on Yan et al.'s method [YLL$^+$09] and the volume version is based on Lévy and Bonneel's method [LB12] with the exact geometric predicates from [Lév16], and then extended in high-d space. All these computations are done under the Euclidean metric, which is easy and efficient, even in the high-d space. In particular, note that the geometric predicates in [Lév16] compute barycentric coordinates, that depend on the intrinsic dimension (2 for surfaces, 3 for volumes) rather than ambient dimension (high-d). Without using the high-d Euclidean embedding, one would need to compute anisotropic RVD and its dual anisotropic mesh on the original surface with the given metric, requiring a high-resolution tessellation of the input surface / volume and additional costly computations, such as computation of anisotropic Voronoi cell boundaries, search for neighboring Voronoi sites in anisotropic metric space, ... Once the RVD is obtained, we can easily compute its dual graph, i.e., Restricted Delaunay Triangulation (RDT) as follows.

Anisotropic RVD and Mesh: Using the barycentric coordinates of each output site or vertex, we can back-project the RVD and RDT from the high-d embedding space onto the original space, and generate the final anisotropic RVD and mesh. As we mentioned before, if the given Riemannian metrics are smooth, the proposed high-d embedding framework can compute a smooth and small-error embedding in higher dimensions. The algorithm robustly computes the RVD using filtered geometric predicates and symbolic perturbation to resolve degeneracies [Lév16]. Note that in general, there is no guarantee that the dual mesh (i.e., the associated RDT) will not have inverted elements (negative Jacobian) when back-projected to 3D. Whenever such an inverted element is detected, our implementation inserts additional points using the provably terminating algorithm in [RLWB16], that uses the same discretization of the metric as us (also used in [LB12]). In practice, we did not observe such inverted

elements in our empirical experiments (the refinement step was never triggered).

5 Evaluations

5.1 Embedding Quality

In order to evaluate the optimized high-d embedding of the surface or volume equipped with a given metric (in practice, the given metric is specified in the input mesh at each vertex), we consider the edge lengths of the input triangular surface or tetrahedral volume under the given metric, and use them as the ground truth to evaluate computed embedding results.

For each edge with vertices $\mathbf{v}_a$ and $\mathbf{v}_b$, we use the average metric of two vertices $\mathbf{Q}_{ab} = (\mathbf{Q}(\mathbf{v}_a) + \mathbf{Q}(\mathbf{v}_b))/2$ to approximate the metric for this edge. Then the length of each edge can be computed according to its own $\mathbf{Q}_{ab}$ and we use it as the ground truth. The *absolute edge length error* is the absolute difference between the edge length of computed embedding and the ground truth. The *relative edge length error* is the percentage of the absolute difference with respect to the ground truth. In our experiments, we only use relative edge length error to measure the computed embedding, because different models may have different sizes and scales. The criteria for evaluating the embedding quality are: L^{rel}_{max} is the maximal value of relative edge length errors of all embedded triangles / tetrahedrons; and L^{rel}_{avg} is the average value of relative edge length errors of all embedded triangles / tetrahedrons. The color-coded diagram of the relative edge length errors is rendered.

5.2 K-NN Efficiency

The following criteria of K-NN efficiency are used: T_{knn} is computational time on K-NN searching for all particles; N_{avg} is the average number of nearest neighbors to be searched; N_{max} is the maximal number of nearest neighbors to be searched. We use the above criteria to compare Zhong et al.'s anisotropic particle optimization [ZGW$^+$13] with our proposed isotropic particle optimization on the high-d Euclidean embedding in Sec 6.2.

5.3 Anisotropic Mesh Quality

Since RVD is not straightforward to be evaluated as compared with its dual triangular and tetrahedral meshes, in the experiments, we will focus on the mesh quality for evaluations.

Surface: In the anisotropic triangular meshing, for each triangle $\triangle_{abc}$ in the final mesh, we use its approximated metric $\mathbf{Q}(\triangle_{abc}) = (\mathbf{Q}(\mathbf{x}_a) + \mathbf{Q}(\mathbf{x}_b) + \mathbf{Q}(\mathbf{x}_c))/3$ to affine-transform it from the original anisotropic space into the Euclidean space. After that, we employ the following isotropic triangular criteria, as suggested by Frey and Borouchaki [FB97] and used in Zhong et al. [ZGW$^+$13] and Fu et al. [FLSG14]'s recent work, to evaluate the quality of generated anisotropic triangular mesh: The quality of a triangle is measured by $G = 2\sqrt{3}\frac{S}{ph}$, where S is the triangle area, p is its half-perimeter, and h is the length

of its longest edge; G_{min} is the minimal quality of all triangles; G_{avg} is the average quality of all triangles; θ_{min} is the smallest angle of the minimal angles of all triangles; θ_{avg} is the average angle of the minimal angles of all triangles; $\%_{<30^\circ}$ is the percentage of triangles with their minimal angles smaller than 30°; The angle histogram is also provided to show the distribution of angles for all triangles. θ_{avg} should be 60° if it is a regular triangle. G is between 0 and 1, where 0 denotes a skinny triangle and 1 denotes a regular triangle.

Volume: In the anisotropic tetrahedral mesh, for each tetrahedron tet_{abcd}, we have its approximated metric $\mathbf{Q}(tet_{abcd}) = (\mathbf{Q}(\mathbf{x}_a) + \mathbf{Q}(\mathbf{x}_b) + \mathbf{Q}(\mathbf{x}_c) + \mathbf{Q}(\mathbf{x}_d))/4$. Then we use the corresponding $\mathbf{Q}(tet_{abcd})$ to affine-transform the tetrahedron in the Euclidean space, and employ the following isotropic tetrahedral mesh quality measurements [DVS+09, YWLL13] to evaluate the quality of generated anisotropic tetrahedral mesh: The quality of a tetrahedron is measured by $G = \frac{12\sqrt[3]{9V^2}}{\sum l_{i,j}^2}$, where V is the volume of the tetrahedron, and $l_{i,j}$ is the length of the edge which connects vertices v_i and v_j. G_{min}, G_{avg} are the minimal and average qualities of all tetrahedrons. θ_{min}, θ_{avg} are the smallest and average angles of the minimal dihedral angles of all tetrahedrons. $\%_{<15^\circ}$ is the percentage of tetrahedrons with their dihedral angles smaller than 15°, which are considered as slivers. The angle histogram is also provided to show the distribution of minimal dihedral angles for all tetrahedrons. θ_{avg} should be $\approx 70.53^\circ$ if it is a regular tetrahedron. G is between 0 and 1, where 0 denotes a sliver and 1 denotes a regular tetrahedron.

6 Results

We implement our algorithms by using Microsoft Visual C++ 2013. The embedding and mesh quality evaluations are implemented with Matlab R2015a. For the hardware platform, the experiments are run on a desktop computer with Intel(R) Core(TM) i7-6700 CPU with 3.40GHz, 32GB DDR4 RAM.

Riemannian Metric Design: In our experiments, users firstly design a smooth scaling field $\mathbf{S}(\mathbf{x})$ and a rotation field $\mathbf{R}(\mathbf{x})$ that is smooth in regions other than some singularities, and then compose them to $\mathbf{M}(\mathbf{x}) = \mathbf{R}(\mathbf{x})^T\mathbf{S}(\mathbf{x})^2\mathbf{R}(\mathbf{x})$, which is similar to the approach mentioned in [DW05a, YS00]. Since this article focuses on both 2- and 3-manifolds, the metric design has some slight differences as follows.

For the anisotropic meshing in 2D domains, we use the following 2×2 metric tensor:

$$\mathbf{M} = [\mathbf{v}_1, \mathbf{v}_2]diag({s_1}^2, {s_2}^2)[\mathbf{v}_1, \mathbf{v}_2]^T, \tag{21}$$

where $\mathbf{v}_1$ and $\mathbf{v}_2$ are two orthogonal principal directions, composing a rotation field. s_1 and s_2 are two user-specified stretching factors along principal directions.

For the anisotropic meshing on 3D surfaces, we use the following 3×3 metric tensor:

$$\mathbf{M} = [\mathbf{v}_{min}, \mathbf{v}_{max}, \mathbf{n}]diag({s_1}^2, {s_2}^2, 0)[\mathbf{v}_{min}, \mathbf{v}_{max}, \mathbf{n}]^T, \tag{22}$$

where $\mathbf{v}_{min}$ and $\mathbf{v}_{max}$ are the directions of the principal curvatures, $\mathbf{n}$ is the unit surface normal. s_1 and s_2 are two user-specified stretching factors along principal curvature directions. Since the surface models are computed by curvature-based metric tensor fields. We use the metric of Eq. (22) with $s_1 = \sqrt{K_{min}}$ and $s_2 = \sqrt{K_{max}}$, where K_{min} and K_{max} are the principal curvatures. We set small thresholds to preserve both K_{min} and K_{max} not vanishing. As suggested by Alliez et al. [ACSD+03], Laplacian smoothing is applied to both the stretching factors and directions, to ensure smoothness of the input metric field on the surface.

For the anisotropic meshing in 3D volumes, we use the following 3×3 metric tensor:

$$\mathbf{M} = [\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3] diag({s_1}^2, {s_2}^2, {s_3}^2)[\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3]^T, \tag{23}$$

where $\mathbf{v}_1$, $\mathbf{v}_2$, and $\mathbf{v}_3$ are three orthogonal principal directions. s_1, s_2, and s_3 are three user-specified stretching factors along principal directions.

6.1 SIFHDE Computation

6.1.1 Importance of Higher Dimensions

In order to demonstrate the importance of the high-d embedding, instead of embedding computed in the original space. We use two examples in detail to explain its advantages.

One example is a 2D domain with a Riemannian metric field. Fig. 2 shows a simple 3D self-intersection free embedding computed by our proposed method for a 2D domain equipped with an anisotropic metric, which can "perfectly" embed the original 2D domain into a Gaussian surface in 3D space. Since we have both the original domain with the metric and its ground truth of the embedded surface (the target dimension is known, i.e., 3D), we can explicitly evaluate our high-d embedding optimization framework both quantitatively and visually.

The pipeline of the proposed work is given in Fig. 2, i.e., to compute the self-intersection free high-d embedding, isotropic particle optimization on such embedding, high-d isotropic RVD and dual meshing, final anisotropic RVD and mesh. The anisotropic metric field is $\mathbf{M}(\mathbf{x}) = \mathbf{R}(\mathbf{x})^T diag\{Stretch(\mathbf{x})^2, 1\}\mathbf{R}(\mathbf{x})$, where the stretching field $Stretch(\mathbf{x}) \in [1, 3.85]$ with the rotation field $\mathbf{R}(\mathbf{x})$. Our proposed embedding optimization framework converges superbly fast, i.e., 20 iterations in this 3D embedding. The embedding quality is quite satisfactory, i.e., $L^{rel}_{avg} = 0.92\%$ and $L^{rel}_{max} = 10.58\%$. Besides the high quality, there is no self-intersection, due to the strategy we discuss in Sec. 3.4.2, i.e., fixing the original 2D coordinates and deforming the new added 3rd coordinates in the embedding computation. Visually, the shape of the embedding result at 20 iterations (Fig. 2 b) looks very similar with the shape of the ground truth Gaussian surface (Fig. 2 a). By using this 3D embedding, we can compute the uniform distribution of particles (Fig. 2 c) for RVD and dual meshing on it (Fig. 2 d), and finally generate high-quality anisotropic RVD and triangular mesh with 2000 output samples as shown in Fig. 2 (e).

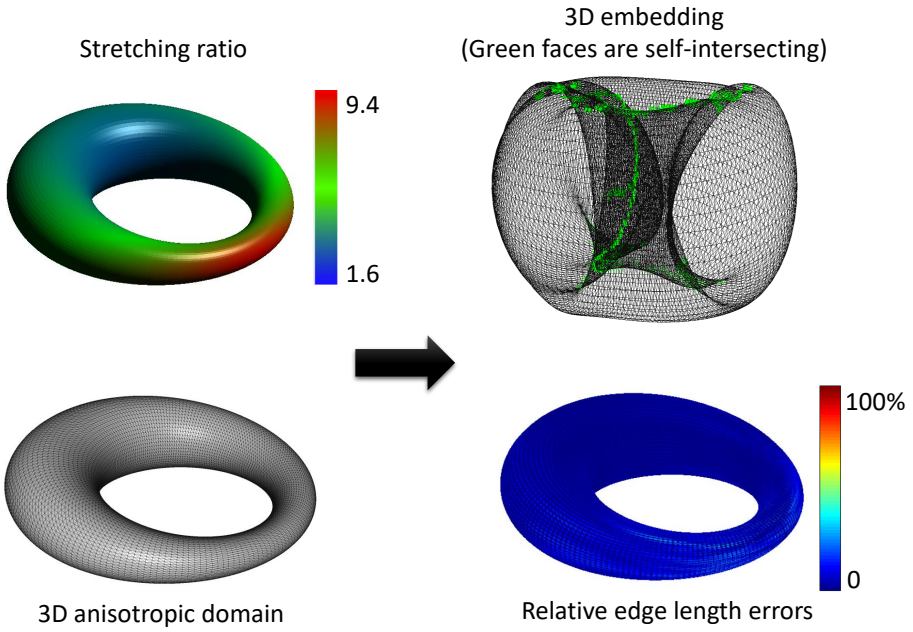

Figure 3: A 3D embedding result of a Cyclide surface with the stretching ratio $\frac{s_2}{s_1} \in [1.6, 9.4]$. There are 1146 self-intersecting faces out of total 21,600 faces as shown in green color.

Note that without using an extra higher dimension (i.e., the 3rd coordinate), the output embedding will have self-intersections, since the input metric controls the deformations of each triangle in the original 2D space, which will very easily cause self-intersections, especially when the metric field has large stretching ratios and directions. Regularizing the embedding to avoid self-intersections is not a good strategy, since it would dramatically increase the error of the embedding either.

Another example is a 3D Cyclide surface with a Riemannian metric field, which is designed according to the surface curvature. The stretching ratio is $\frac{s_2}{s_1} \in [1.6, 9.4]$. Without a high-d embedding, i.e., using a 3D embedding as suggested by Panozzo et al. [PPTSH14], in Fig. 3, though the embedding errors are relatively small, i.e., $L^{rel}_{avg} = 2.36\%$ and $L^{rel}_{max} = 30.62\%$ (still larger than our 8D embedding errors as shown in Fig. 4), there are many self-intersecting faces in the computed 3D embedding (i.e., 1146 self-intersecting faces in this Cyclide model), which are harmful for the particle optimization, RVD and dual mesh computations. The major fundamental difference between their framework and ours is that they deform the embedding directly in the original 3D coordinates, which may lead self-intersections; however, we deform the embedding in high-d space with fixing the first three coordinates to avoid self-intersections (as discussed in Sec. 3.4.2).

Table 1: Statistics of self-intersecting faces for embeddings in 3D and high-d spaces on different surface models.

Model	Cyclide1	Cyclide2	Kitten	Vase	Knot	Club	Rocker Arm	Hand
$3D$	1146	1751	1001	502	6552	4444	7545	616
High-D	0	0	0	0	0	0	0	0

Furthermore, we also investigate some other surface models and Tab. 1 shows that self-intersecting faces do widely exist in the 3D embedding (i.e., Panozzo et al.'s method [PPTSH14]); however, our high-d embedding framework does not have such problem. Due to the page limit, we do not provide the quantitative statistics of self-intersecting faces in 2D domain and 3D volume models, but it is clear that the same conclusion can be drawn.

6.1.2 Choosing the Dimension of the Embedding

We elaborate on how to choose the dimension of the high-d target for computing the SIFHDE with small relative edge length errors. In general, we do not know beforehand the required dimension for the high-d embedding. From Nash theorem, it is known that for constructing a C^1 isometric embedding of a nD metric, $2n$ dimensions suffice (Nash-Kuiper). For higher smoothness ($C^k, k \geq 2$), the bound on embedding dimension becomes $n(3n + 11)/2$ (Nash-Moser). With our regularization term, we constrain the embedding to be C^2 and thus avoid the oscillations of C^1 embeddings, but the Nash-Moser theoretical bound becomes 17D for surfaces and 30D for volumes. We now evaluate the required embedding dimension in practice for surfaces and volumes and show empirically that in both cases 8D suffices to accurately account for the input metric:

Surface Case: We conduct different experiments with surfaces. The associated metric deviation errors of a Cyclide surface are plotted in Fig. 4. It is noted that in our case, when using embedding dimensions larger than 8D, the relative edge length errors are quite stable and small.

Furthermore, we also investigate some other surface models (as shown in Fig. 5) to conclude that 8D space is a good target dimension for computing SIFHDE with smaller relative edge length errors, e.g., Nefertiti surface: $L^{rel}_{avg} = 2.98\%$ and $L^{rel}_{max} = 46.33\%$, Kitten surface: $L^{rel}_{avg} = 3.82\%$ and $L^{rel}_{max} = 139.21\%$.

Volume Case: As for the volume models, we also investigate the best target dimension for computing the self-intersection free embedding with smaller relative edge length errors. Similar to surface case, volume models (as shown in Fig. 6) also conclude that 8D is a good target dimension for computing SIFHDE with smaller relative edge length errors, e.g., Cube volume: $L^{rel}_{avg} = 1.65\%$ and $L^{rel}_{max} = 17.54\%$, Sphere volume: $L^{rel}_{avg} = 3.21\%$ and $L^{rel}_{max} = 65.01\%$. As in the surface case, we also observe that beyond 8D, the relative edge length errors remain stable and small.

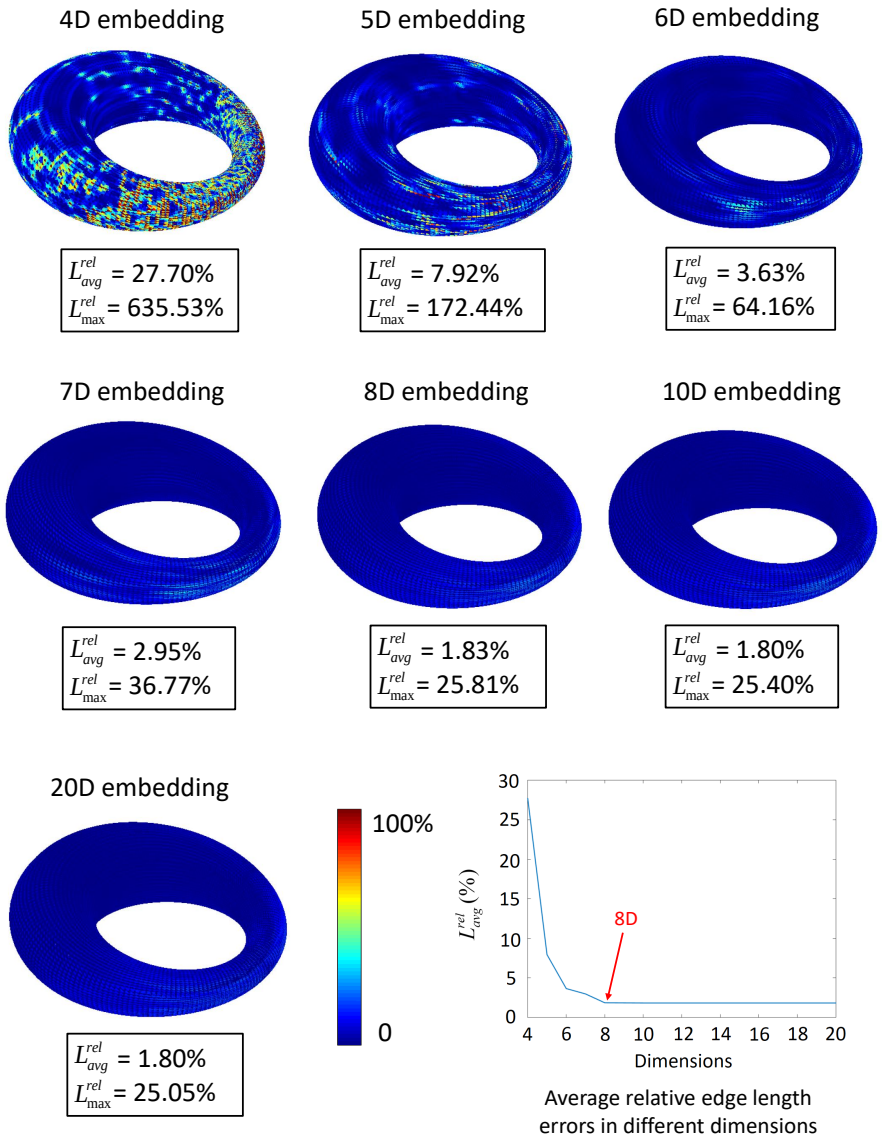

Figure 4: The color-coded relative edge length error images of high-d embeddings (i.e., $4D \sim 20D$) of the Cyclide surface.

6.2 K-NN Efficiency on High-D Embedding

In order to compare the efficiency of the K-NN between the high-d embedding space and the original space, we use the same search radius 5σ, where σ is the kernel width of the inter-particle energy in the Euclidean embedding space

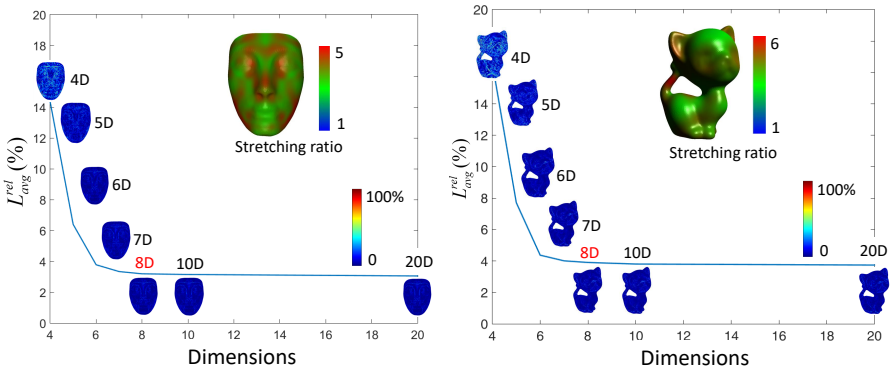

Figure 5: The relative edge length errors of high-d embeddings (i.e., $4D \sim 20D$) of the Nefertiti and Kitten surfaces.

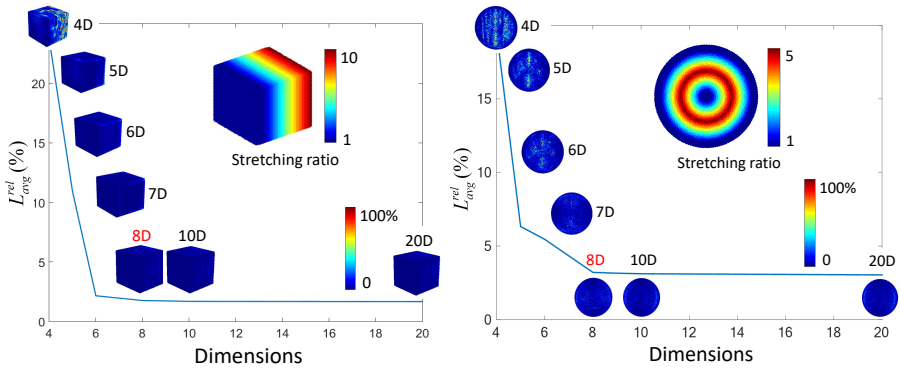

Figure 6: The relative edge length errors of high-d embeddings (i.e., $4D \sim 20D$) of the Cube and Sphere (shown its cross section) volumes.

(as mentioned in Sec. 4.1). However, when the K-NN computation happens in the original space, we need to transform this search radius into the original anisotropic space, resulting in a large range once the anisotropic stretching ratio is high. After that, we need to further check and prune the spurious neighbors under the given metric.

Fig. 7 shows the K-NN computation of 2000 particles optimization on the computed 8D Euclidean embedding in this article and the original 3D surface [ZGW+13] of the Cyclide model with the stretching ratio $\frac{s_2}{s_1} \in [1.6, 9.4]$. We provide the statistics when the particles are at equilibrium state, i.e., at the end of the optimization, since the number of neighbors may vary slightly at each iteration during optimization. The K-NN at equilibrium state is more convincing and stable to analyze. It demonstrates that the average K-NN on 8D Euclidean embedding is 19 compared with 103 on the original anisotropic space, and the K-NN searching time of 2000 particles for each iteration on the embedded surface is about 5 times faster than that of the original surface with the specified anisotropic metric.

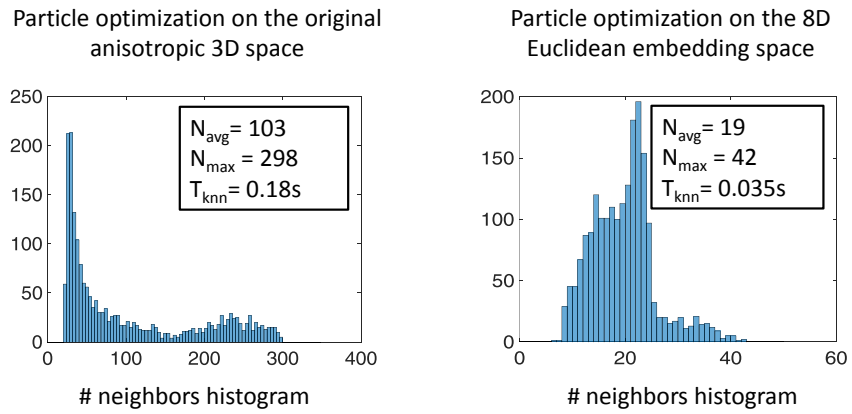

Figure 7: Comparison of K-NN searching of 2000 particles optimization on the original anisotropic 3D Cyclide surface and its 8D Euclidean embedded surface.

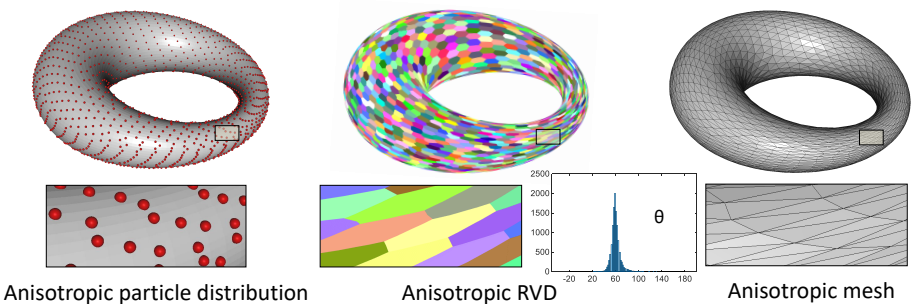

Figure 8: The anisotropic particle distribution, RVD, and its dual anisotropic mesh of the Cyclide surface by using the 8D embedding.

Based on the optimized particles on the self-intersection free 8D embedding, we can compute the RVD and its dual mesh, and finally map them back to the original 3D space to generate the anisotropic RVD and its dual anisotropic mesh. The results of the Cyclide model are shown in Fig. 8.

Besides Cyclide model, we also measure the K-NN efficiency on the high-d embeddings of other models, such as Kitten, Vase, Knot, Club, etc., and the average K-NN is quite stable and small, i.e., $N_{avg} = 20$. However, without using Euclidean embedding, the average K-NN highly depends on the stretching ratio in the Riemannian metric. The higher the stretching ratio is, the larger the average K-NN is, such as $N_{avg} = 232$ on another Cyclide model (as shown in Fig. 12) with the larger stretching ratio $\frac{s_2}{s_1} \in [2, 29]$.

6.3 2D RVD and Meshing

A 2D domain can be considered as a 3D flat surface, so 8D is a good dimension for the embedding computation. Besides the previous 2D domain with Gaussian embedding metric in Fig. 2, we also evaluate our method on the largely varying anisotropic field. Fig. 9 shows the 8D embedding errors ($L^{rel}_{avg} = 4.04\%$ and $L^{rel}_{max} = 76.79\%$), the anisotropic RVD, and meshing results of a 2D square domain with 4000 output samples, with a highly-varying metric field $\mathbf{M}(\mathbf{x}) = \mathbf{R}(\mathbf{x})^T diag\{Stretch(\mathbf{x})^2, 1\}\mathbf{R}(\mathbf{x})$, where the stretching field $Stretch(\mathbf{x}) \in [1, 40]$ with the rotation field $\mathbf{R}(\mathbf{x})$, which is defined by the Hessian of the non-convex analytic functions $u(x, y) = tanh(10(sin(5y) - 2x)) + x^2y + y^3$.

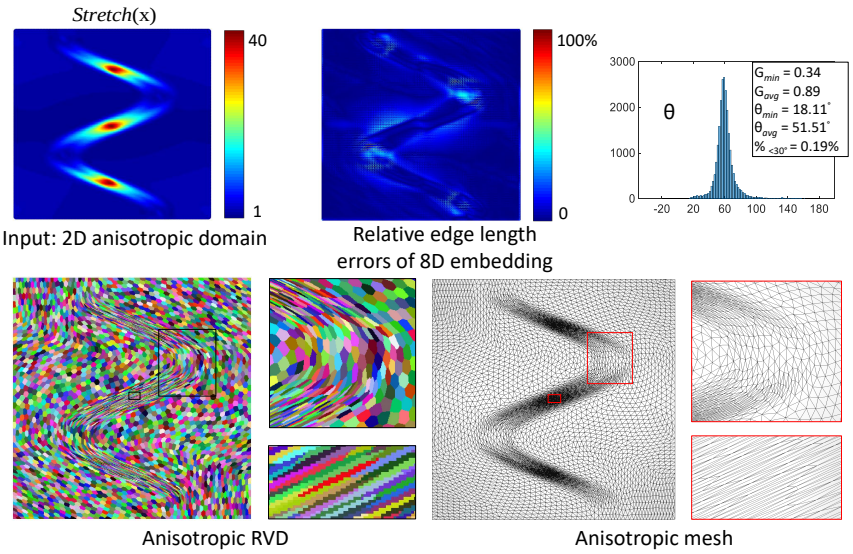

Figure 9: The embedding errors, anisotropic RVD, and anisotropic meshing results of a 2D domain with a highly-varying metric field $\mathbf{M}(\mathbf{x}) = diag\{Stretch(\mathbf{x})^2, 1\}$, where $Stretch(\mathbf{x}) \in [1, 40]$.

6.4 3D Surface RVD and Meshing

Our Method Results: Fig. 10 shows the anisotropic 3D surface RVD and meshing results of Kitten, Vase, Knot, Club, Rocker Arm, and Hand models with the anisotropic metrics designed by the surfaces' curvature tensors. The statistics and timings for our 8D embedding computation and surface meshing are shown in Tab. 2. It is noted that our embedding computation in 8D space is quite efficient, and all surface models provided in this article need only dozens of seconds.

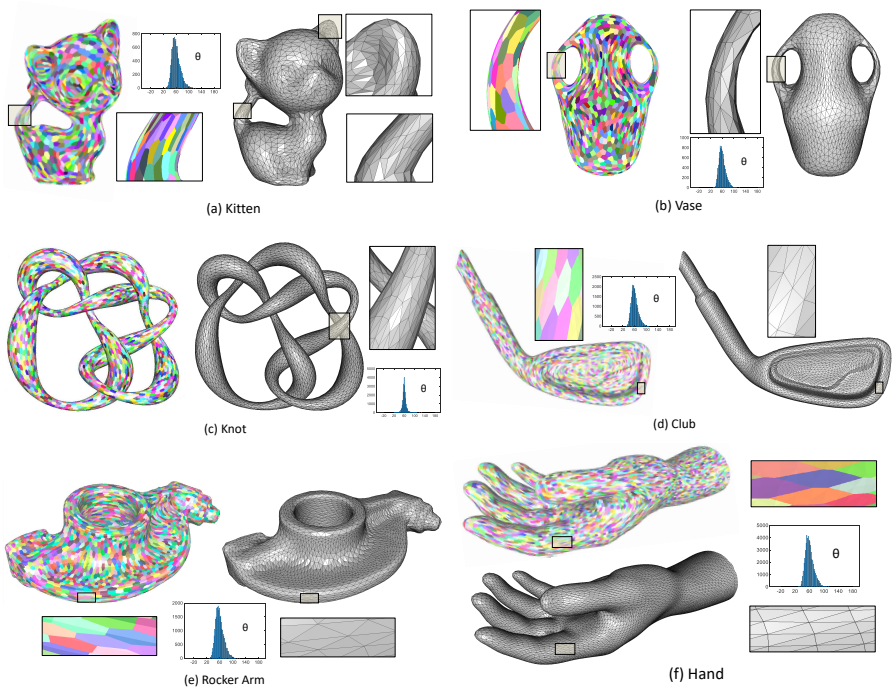

Figure 10: The anisotropic 3D surface RVD and meshes (Kitten, Vase, Knot, Club, Rocker Arm and Hand models) generated by our 8D embedding method.

Table 2: Statistics and timings for embedding computation and surface meshing.

Model	In # V.	Stretch	L^{rel}_{avg}	L^{rel}_{max}	Embed. (s)	Out # V.	G_{min}	G_{avg}	θ_{min}	θ_{avg}	$\%_{<30^\circ}$	Mesh (s)
Cyclide1 (Fig. 8)	10,800	[1.6, 9.4]	1.83%	25.81%	9.14	2000	0.57	0.92	28.88°	54.47°	0.05%	6.22
Cyclide2 (Fig. 12)	25,920	[2, 29]	2.18%	19.39%	23.63	8000	0.63	0.92	29.07°	53.70°	0.009%	42.83
Kitten	20,000	[1, 6]	3.82%	139.21%	17.56	2000	0.33	0.85	20.69°	48.60°	0.34%	7.87
Vase	20,000	[1, 5]	4.18%	157.97%	17.09	2000	0.36	0.86	21.33°	49.52°	0.24%	7.48
Knot	24,392	[2, 8]	4.49%	168.46%	21.16	5000	0.47	0.93	25.37°	53.79°	0.05%	26.41
Club	30,000	[1, 6]	5.44%	143.44%	27.82	5000	0.46	0.91	24.10°	51.25°	0.07%	29.94
Rocker Arm	35,840	[1, 7]	5.95%	138.31%	34.77	5000	0.37	0.86	19.59°	47.65°	0.16%	33.39
Hand	36,619	[1, 5]	2.68%	121.57%	35.78	10,000	0.28	0.90	16.28°	50.14°	0.23%	55.50

Note: Embed. Time: timing for embedding computation with 50 iterations. Mesh Time: timing for both particle optimization with 50 iterations and RVD computation.

Comparison with Other Embedding Methods: Fig. 11 shows the comparison with 2D conformal embedding method [ZSJG14] and Lévy and Bonneel's 6D embedding [LB12], and our method, on a Kitten surface model with 5000 output samples. The 2D conformal embedding method produces poor meshing result, since it computes the embedded surface into 2D space, which has very limited degrees of freedom. However, the 6D embedding has more degrees of freedom, but it applies a 6D metric in terms of vertex positions and normals,

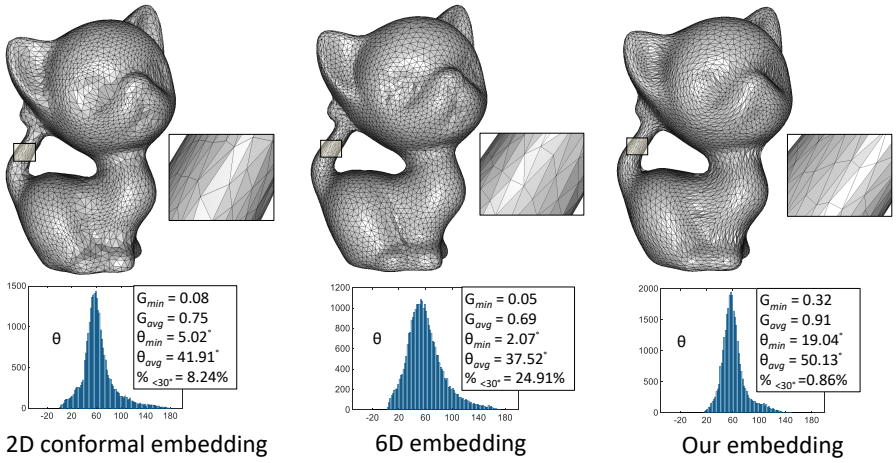

Figure 11: Comparison on anisotropic meshing results with 2D conformal embedding method [ZSJG14], Lévy and Bonneel's 6D embedding [LB12], and our method on the Kitten surface model.

without considering the curvature tensors, so its meshing result is even worse than 2D conformal embedding. Our high-d embedding method, i.e., 8D, obtains better mesh quality result and it also better matches the input curvature-based metric field. We also use Hausdorff distance to measure the faithfulness accuracy to the original input surface of these three methods. The Hausdorff distance of our result is 0.004964% of the bounding-box's diagonal length, while the Hausdorff distances of 2D conformal embedding and 6D embedding results are 0.013415% and 0.005110%, respectively. It is noted that the meshing fidelity of 2D embedding is worse than both those of 6D and our 8D embeddings, though it has a reasonable mesh quality result to match the designed metric. Essentially, when using dimension reduction strategy, such as mapping a 3D shape into a 2D domain, it may lose some 3D shape fidelity information.

We also try to compare with the 3D embedding method [PPTSH14], but there are 1001 self-intersecting faces in the Kitten model as shown in Tab. 1. Then, if we would like to compute RVD and meshing on it, some parameterization methods are needed, which is beyond the scope of this article. Due to this limitation, we do not provide the quantitative comparison between the 3D embedding and our method for meshing computation.

Comparison with Other Anisotropic Surface Meshing Methods: In this subsection, we show further comparative analysis and experiments with other state-of-the-art anisotropic meshing approaches. Fig. 12 compares our method with both particle-based method [ZGW+13] and LCT method [FLSG14] on a Cyclide model with the large stretching ratio $\frac{s_2}{s_1} \in [2, 29]$. Our method improves the mesh quality significantly as shown in Fig. 12 compared with the original particle-based method [ZGW+13], since we have computed the particle optimization directly on the high-d embedding. Compared with LCT method (one of the optimal high-quality anisotropic meshing methods), our method

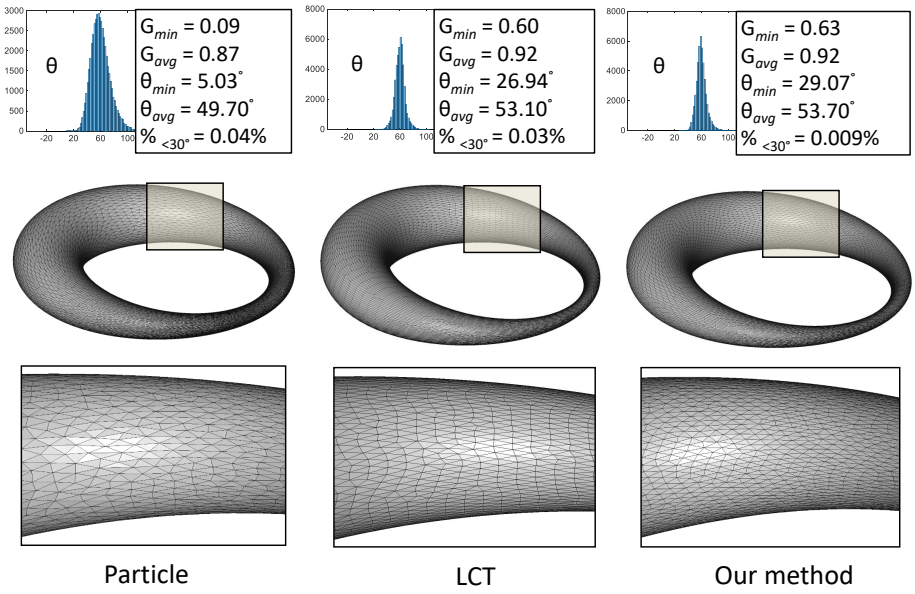

Figure 12: Comparison on anisotropic surface meshing results (8000 output samples) with particle-based method [ZGW+13], LCT method [FLSG14], and our method on the Cyclide surface model with the stretching ratio $\frac{s_2}{s_1} \in [2, 29]$.

increases the minimal angle from 26.94° to 29.07° and $\%_{<30^\circ} = 0.03\%$ versus $\%_{<30^\circ} = 0.009\%$, meanwhile we have similar quality of average values. Fig. 13 visualizes the final mesh quality errors (i.e., 1-G, where G is the quality of triangle as mentioned in Sec. 5.3) of the above three methods. We can see that our errors are more smoothly distributed, whereas both particle-based method and LCT method have more poor quality triangles.

6.5 3D Volume RVD and Meshing

The anisotropic metric fields in 3D volume models are designed by different analytic functions in our experiments. The statistics and timings for our 8D embedding computation and volume meshing are shown in Tab. 3. Fig. 14 shows the anisotropic 3D volume RVD and meshing results of a Cube model with domain $[0, 1]^3$ of two different Riemannian metric fields $\mathbf{M}(\mathbf{x}) = diag\{Stretch(\mathbf{x})^2, 1, 1\}$. The first one is designed by a linear function with stretching ratios $Stretch(\mathbf{x}) = 10x$, where $Stretch(\mathbf{x}) \in [1, 10]$, and the second one is designed by a highly nonlinear function with large stretching ratios $Stretch(\mathbf{x}) = 0.2(0.0025 + 0.2(1 - e^{-|x-0.5|}))^{-0.8}, \in [1, 25]$.

A unit Sphere model with a cylindrical anisotropic metric field is given in Fig. 1. The Riemannian metric is designed as $\mathbf{M}(\mathbf{x})$=$\mathbf{R}(\mathbf{x})^T diag\{Stretch(\mathbf{x})^2, 1, 1\}\mathbf{R}(\mathbf{x})$, where $Stretch(\mathbf{x}) \in [1, 5]$, and $\mathbf{R}(\mathbf{x})$'s three columns are $(x/\sqrt{x^2+y^2}, y/\sqrt{x^2+y^2}, 0)^T$, $(-y/\sqrt{x^2+y^2}, x/\sqrt{x^2+y^2}, 0)^T$, and $(0, 0, 1)^T$.

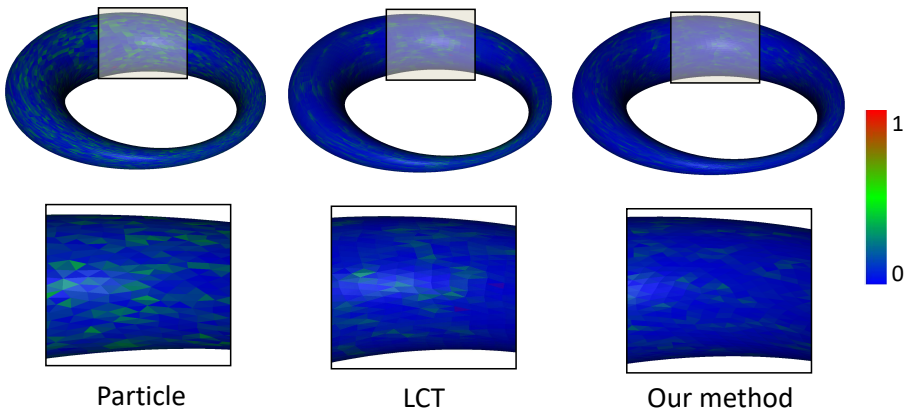

Figure 13: Comparison on final mesh quality errors (i.e., 1-G) with particle-based method [ZGW+13], LCT method [FLSG14], and our method on the Cyclide surface model with the stretching ratio $\frac{s_2}{s_*} \in [2, 29]$.

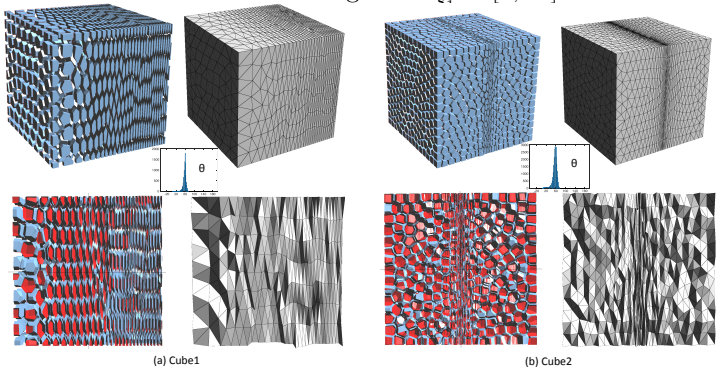

Figure 14: The anisotropic 3D volume RVD and meshes on two Cube models (with different Riemannian metric fields) generated by our 8D embedding method. Both the surface and interior of the volume RVD and tetrahedral meshes are shown. In order to better visualize the 3D RVD results, we shrink each Voronoi cell (i.e., polyhedron) a little bit. The blue color represents the outside of each 3D Voronoi cell, and the red color represents the inside of each 3D Voronoi cell.

Fig. 15 shows one volumetric RVD result on a Cube model with domain $[1, 11]^3$. The targeted cylindrical Riemannian metric field is specified via a highly nonlinear analytic function as $\mathbf{M}(\mathbf{x}) = \mathbf{R}(\mathbf{x})^T diag\{Stretch(\mathbf{x})^2, 1, 1\}\mathbf{R}(\mathbf{x})$, where $Stretch(\mathbf{x}) = 2(0.1 + 2(1 - e^{-0.01|x^2+y^2-49|}))^{-1}$, and $\mathbf{R}(\mathbf{x})$ is a cylindrical rotation field as defined previously. The stretching ratio is $Stretch(\mathbf{x}) \in [1, 20]$. By using our method, we can generate highly curved 3D Voronoi cells in regions where the metric varies quickly, while in low stretching regions, the regular 3D

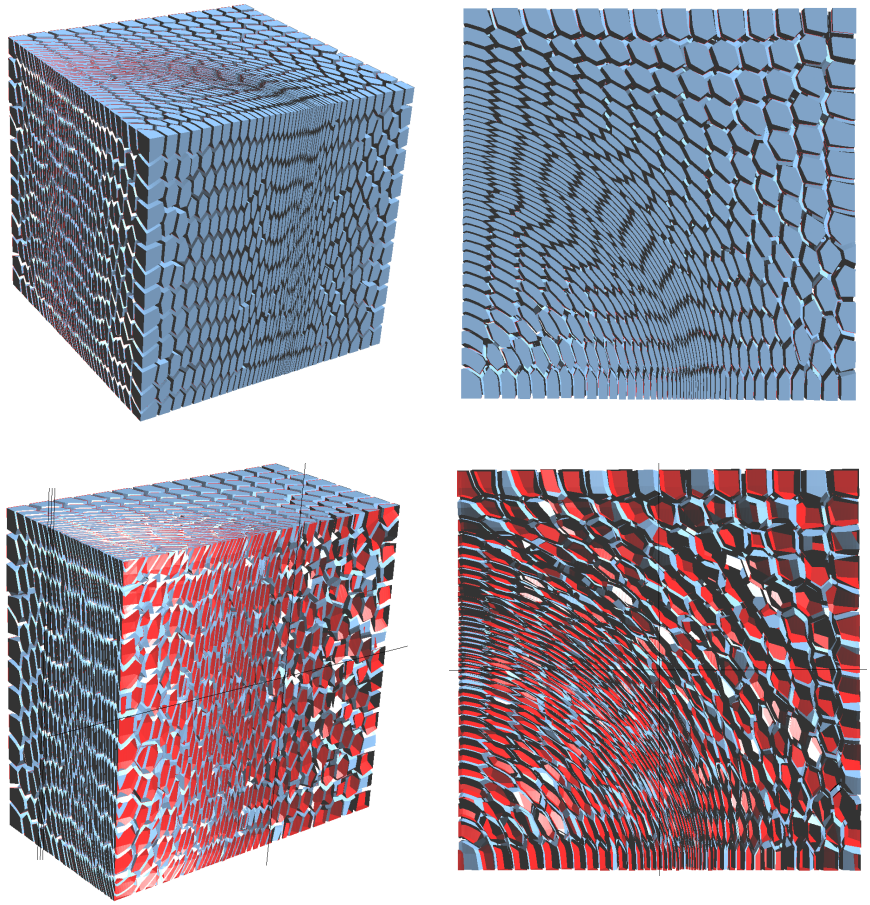

Figure 15: The anisotropic 3D volume RVD on the Cube model with the large cylindrical anisotropic metric field generated by our 8D embedding method. Both the surface and interior of the volume RVD are shown.

Voronoi cells are obtained. Its dual tetrahedral mesh is given in the following section.

Fig. 16 shows the meshing comparison with our method and LCT method [FLSG14] on a Cube model with the previous cylindrical Riemannian metric field. By using almost the same number of vertices, i.e., 6338 vertices in LCT method and 6300 vertices in our method, we achieve better mesh quality results in average value measurements as shown in Fig. 16. Visually, our result achieves better mesh stretching ratios and directions in both surface and interior of the volume. The reason why we have worse results in minimal mesh quality and smallest angle is we do not apply any sliver elimination strategy in our meshing method. Since the sliver elimination is a post-processing in both methods, it is more straightforward to compare the performance of two algorithms without doing

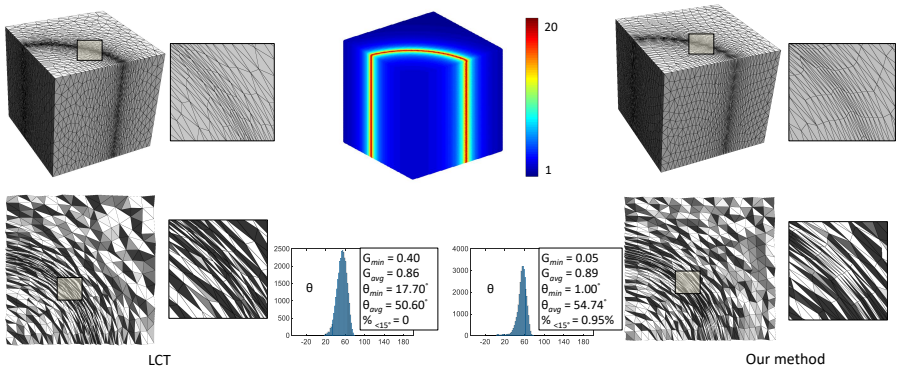

Figure 16: Comparison on anisotropic volume meshing results with LCT method [FLSG14] and our method on the Cube volume model with the cylindrical Riemannian metric field of stretching ratio $Stretch(\mathbf{x}) \in [1, 20]$.

Table 3: Statistics and timings for embedding computation and volume meshing.

Model	In # V.	Stretch	L^{rel}_{avg}	L^{rel}_{max}	Embed. (s)	Out # V.	G_{min}	G_{avg}	θ_{min}	θ_{avg}	$\%_{<15^\circ}$	Mesh (s)
Cube1 (Fig. 14 a)	9261	[1, 10]	1.65%	17.54%	138.84	2000	0.09	0.92	1.66°	57.24°	0.29%	29.71
Cube2 (Fig. 14 b)	44, 541	[1, 25]	1.30%	48.39%	567.56	5000	0.10	0.91	1.68°	56.76°	0.42%	83.73
Cube3 (Fig. 16)	44, 541	[1, 20]	2.12%	71.32%	562.28	6300	0.05	0.89	1.0°	54.74°	0.95%	96.92
Sphere (Fig. 1)	50, 000	[1, 5]	3.21%	65.01%	587.61	5000	0.08	0.88	1.34°	54.02°	1.24%	77.67

Note: Embed. Time: timing for embedding computation with 50 iterations. Mesh Time: timing for both particle optimization with 50 iterations and RVD computation. The small values are in G_{min} and θ_{min}, since we do not apply any sliver elimination strategy (a post-processing) in our meshing results.

sliver elimination. As mentioned in [FLSG14], they have 517 slivers (i.e., tetrahedrons with the dihedral angle $< 15\,^\circ$) before applying sliver elimination, and we have a smaller number of slivers, i.e., 311. It is noted that the LCT result in Fig. 16 is after applying sliver elimination, however, our result is not.

Lastly, we show one more volumetric RVD result on a unit Sphere model with a smaller number of samples, i.e., 1000, in Fig. 17. The Riemannian metric is defined as a cosine wave anisotropy: $\mathbf{M}(\mathbf{x}) = \mathbf{R}(\mathbf{x})^T diag\{Stretch(\mathbf{x})^2, 1, 1\}\mathbf{R}(\mathbf{x})$, where $Stretch(\mathbf{x}) = 10$, and $\mathbf{R}(\mathbf{x})$'s three columns are $(2cos(4x), 1, 0)^T$, $(1, -2cos(4x), 0)^T$, and $(0, 0, 1)^T$. A closeup shows clearly how the 3D Voronoi cells are extremely curved. Both the surface and interior of the volumetric RVD are illustrated. This example is to demonstrate the advantage of using our proposed SIFHDE as a tool to compute arbitrary anisotropic Voronoi diagrams. Without using SIFHDE, such as Budninskiy et al. [BLdG$^+$16]'s work by using a convex function to represent the anisotropy, it is very difficult to obtain our result.

Through the above experiments, it is noted that our proposed SIFHDE and high-d RVD are effective computational algorithms to generate the high-quality anisotropic 3D Voronoi diagrams and tetrahedral meshes with arbitrary smooth Riemannian metric fields.

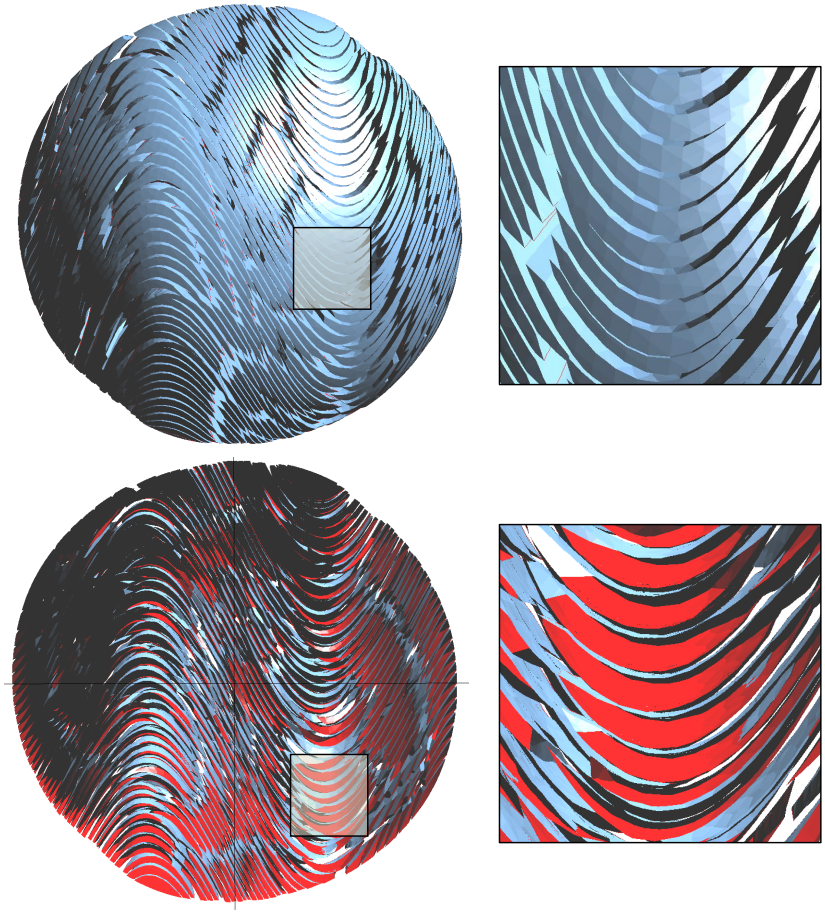

Figure 17: The anisotropic 3D volume RVD on the Sphere model (with a cosine wave anisotropy) generated by our 8D embedding method. Both the surface and interior of the volume RVD are shown.

7 Discussion and Future Work

To the best of our knowledge, this is the first article in the literature for computing the self-intersection free Euclidean embedding in arbitrary dimensions and using it in Voronoi Diagram, surfaces and volume meshing equipped with Riemannian metrics. In particular, this is the first practical algorithm for computing volumetric anisotropic Voronoi diagrams. While there is a direct application to anisotropic meshing, we feel that there are many unexplored applications as limitations and future work: (1) We have not yet explored the cases where the metric has sharp discontinuities, in such case, finding a practical embedding may be challenging. (2) In this work, we would like to emphasize that all the costs of computations in a high-d space are justified since we can obtain high-quality anisotropic RVD and meshes as compared with previous methods. In the future,

we will improve the computational speed by using GPU-based parallel algorithm and implementation. (3) In order to generate the sliver-free tetrahedral meshes, we will apply some sliver suppressing strategies, such as perturbations [TSA09], sliver elimination [FLSG14] or gradient-based shape matching [NZL$^+$17]. (4) Besides anisotropic RVD and meshing, we will explore other important and interesting applications by using the proposed SIFHDE.

References

[ACSD$^+$03] P. Alliez, D. Cohen-Steiner, O. Devillers, B. Lévy, and M. Desbrun. Anisotropic polygonal remeshing. *ACM Transactions on Graphics*, 22(3):485–493, 2003.

[ACSYD05] Pierre Alliez, David Cohen-Steiner, Mariette Yvinec, and Mathieu Desbrun. Variational tetrahedral meshing. *ACM Transactions on Graphics*, 24(3):617–625, 2005.

[AL10] F. Alauzet and A. Loseille. High-order sonic boom modeling based on adaptive methods. *Journal of Computational Physics*, 229(3):561–593, 2010.

[AL16] F. Alauzet and A. Lozeille. A decade of progress on anisotropic mesh adaptation for computational fluid dynamics. *Computer-Aided Design*, 72:13–39, 2016.

[BBKY00] M. M. Bronstein, A. M. Bronstein, R. Kimmel, and I. Yavneh. Multigrid multidimensional scaling. *Numer. Linear Algebra Appl.*, 0:1–6, 2000.

[BBKY05] M. M. Bronstein, A. M. Bronstein, R. Kimmel, and I. Yavneh. A multigrid approach for multi-dimensional scaling. In *Proceedings of the Copper Mountain Conference on Multigrid Methods*, 2005.

[BGH$^+$97] H. Borouchaki, P. L. George, F. Hecht, P. Laug, and E. Saltel. Delaunay mesh generation governed by metric specifications. part I. algorithms. *Finite Elements in Analysis and Design*, 25(1–2):61–83, 1997.

[BGM97] H. Borouchaki, P. L. George, and B. Mohammadi. Delaunay mesh generation governed by metric specifications. part II. applications. *Finite Elements in Analysis and Design*, 25(1–2):85–109, 1997.

[BH96] F. Bossen and P. Heckbert. A pliant method for anisotropic mesh generation. In *5th International Meshing Roundtable*, pages 63–76, 1996.

[BJLT12] Vincent Borelli, Said Jabrane, Francis Lazarus, and Boris Thibert. Flat tori in three-dimensional space and convex integration. *Proceedings of the National Academy of Sciences of the United States of America*, 109(19), 2012.

[BLdG+16] Max Budninskiy, Beibei Liu, Fernando de Goes, Yiying Tong, Pierre Alliez, and Mathieu Desbrun. Optimal Voronoi tessellations with hessian-based anisotropy. *ACM Transactions on Graphics*, 35(6):242:1–242:12, 2016.

[BSTY14] Jean-Daniel Boissonnat, Kan-Le Shi, Jane Tournois, and Mariette Yvinec. Anisotropic Delaunay meshes of surfaces. *ACM Trans. Graph.*, 32(4), 2014.

[BWY08a] J. Boissonnat, C. Wormser, and M. Yvinec. Anisotropic diagrams: Labelle shewchuk approach revisited. *Theoretical Computer Science*, 408(2-3):163–173, 2008.

[BWY08b] J. Boissonnat, C. Wormser, and M. Yvinec. Locally uniform anisotropic meshing. In *Proceedings of the 24th annual symposium on Computational geometry*, SCG '08, pages 270–277, 2008.

[BWY15] J. Boissonnat, C. Wormser, and M. Yvinec. Anisotropic Delaunay mesh generation. *SIAM J. Comput.*, 44(2):467–512, 2015.

[CDES01] H-L. Cheng, T. Dey, H. Edelsbrunner, and J. Sullivan. Dynamic skin triangulation. *ACM-SIAM symposium on Discrete algorithms*, 25:525–568, 2001.

[CDR06] S. Cheng, T. Dey, and E. Ramos. Anisotropic surface meshing. In *Proc. ACM-SIAM Sympos. Discrete Algorithms*, pages 202–211, 2006.

[CDRR04] S. Cheng, T. Dey, E. Ramos, and T. Ray. Sampling and meshing a surface with guaranteed topology and geometry. In *Proc. 20th Sympos. Comput. Geom.*, pages 280–289, 2004.

[CLGD06] F. Courty, D. Leservoisier, P.-L. George, and A. Dervieux. Continuous metric and mesh adaptation. *Applied Numerical Mathematics*, 55:117–145, 2006.

[CnG06] Guillermo D. Cañas and Steven J. Gortler. Surface remeshing in arbitrary codimensions. *Visual Computer*, 22(9):885–895, 2006.

[CSX07] L. Chen, P. Sun, and J. Xu. Optimal anisotropic meshes for minimizing interpolation errors in L^p-norm. *Math. Comp.*, 76:179–204, 2007.

[CWL+14] Zhonggui Chen, Wenping Wang, Bruno Lévy, Ligang Liu, and Feng Sun. Revisiting optimal delaunay triangulation for 3d graded mesh generation. *SIAM Journal Scientific Computing*, 2014.

[CX04] L. Chen and J. Xu. Optimal Delaunay triangulations. *Journal of Computational Mathematics*, 22:299–308, 2004.

[DF08] C. Dobrzynski and P. Frey. Anisotropic Delaunay mesh adaptation for unsteady simulations. In *17th International Meshing Roundtable*, pages 177–194, 2008.

[DFG99] Q. Du, V. Faber, and M. Gunzburger. Centroidal Voronoi tessellations: Applications and algorithms. *SIAM Review*, 41(4):637–676, 1999.

[DMS14] F. Dassi, A. Mola, and H. Si. Curvature-adapted remeshing of CAD surfaces. In *23rd International Meshing Roundtable*, pages 253–265, 2014.

[DSPS15] F. Dassi, H. Si, S. Perotto, and T. Streckenbach. Anisotropic finite element mesh adaptation via higher dimensional embedding. In *24th International Meshing Roundtable*, pages 265–277, 2015.

[DVS+09] J. Dardenne, S. Valette, N. Siauve, N. Burais, and R. Prost. Variational tetrahedral mesh generation from discrete volume data. *The Visual Computer*, 25(5):401–410, 2009.

[DW05a] Q. Du and D. Wang. Anisotropic centroidal Voronoi tessellations and their applications. *SIAM Journal on Scientific Computing*, 26(3):737–761, 2005.

[DW05b] Qiang Du and Desheng Wang. The optimal centroidal Voronoi tessellations and the gersho's conjecture in the three-dimensional space. *Computers & Mathematics with Applications*, 49(9):1355–1373, 2005.

[FA05] P. Frey and F. Alauzet. Anisotropic mesh adaptation for CFD computations. *Comp. Methods in App. Mechanics and Eng.*, 194(48-49):5068–5082, 2005.

[FB97] P. J. Frey and H. Borouchaki. Surface mesh evaluation. In *6th International Meshing Roundtable*, pages 363–373, 1997.

[FLSG14] X. Fu, Y. Liu, J. Snyder, and B. Guo. Anisotropic simplicial meshing using local convex functions. *ACM Transactions on Graphics*, 33(6):182:1–182:11, 2014.

[GL96] G. H. Golub and C. F. V. Loan. *Matrix Computations, third ed.* Johns Hopkins University Press, Baltimore, Maryland, 1996.

[Kli08] Bryan Matthew Klingner. *Tetrahedral Mesh Improvement.* Ph.D. Thesis, Dep. of Elec. Eng. and Comp. Sciences U. of California at Berkeley, 2008.

[KMZ10] Denis Kovacs, Ashish Myles, and Denis Zorin. Anisotropic quadrangulation. In *Proceedings of the 14th ACM Symposium on Solid and Physical Modeling*, SPM '10, pages 137–146, 2010.

[KS07] Bryan Matthew Klingner and Jonathan Richard Shewchuk. Agressive tetrahedral mesh improvement. In *Proceedings of 16th International Meshing Roundtable*, pages 3–23, 2007.

[Kui55] N. H. Kuiper. On C^1-isometric embeddings I. In *Proc. Nederl. Akad. Wetensch. Ser. A*, pages 545–556, 1955.

[LB12] B. Lévy and N. Bonneel. Variational anisotropic surface meshing with Voronoi parallel linear enumeration. In *21st International Meshing Roundtable*, pages 349–366, 2012.

[Lév16] Bruno Lévy. Robustness and efficiency of geometric programs: The predicate construction kit (PCK). *Computer-Aided Design*, 72:3–12, 2016.

[LL16] A. Loseille and R. Löhner. Anisotropic mesh generation for high-fidelity simulations in CFD. *INRIA*, 2016. preprint.

[Llo82] S. Lloyd. Least squares quantization in PCM. *IEEE Transactions on Information Theory*, 28(2):129–137, 1982.

[LN89] D. C. Liu and J. Nocedal. On the limited memory BFGS method for large scale optimization. *Mathematical Programming*, 45(3):503–528, 1989.

[LWL+09] Y. Liu, W. Wang, B. Lévy, F. Sun, D. Yan, L. Lu, and C. Yang. On centroidal Voronoi tessellation – energy smoothness and fast computation. *ACM Transactions on Graphics*, 28(4):101:1–101:17, 2009.

[MGR13] E. Marchandise, C. Geuzaine, and J.F. Remacle. Cardiovascular and lung mesh generation based on centerlines. *International Journal for Numerical Methods in Biomedical Engineering*, 29(6):665–682, 2013.

[Nas54] J. Nash. C^1-isometric embeddings. *Annals of Mathematics*, 60(3):383–396, 1954.

[NSO12] R. Narain, A. Samii, and J. F. O'Brien. Adaptive anisotropic remeshing for cloth simulation. *ACM Transactions on Graphics*, 31(6):147:1–147:10, 2012.

[NZL+17] Saifeng Ni, Zichun Zhong, Yang Liu, Wenping Wang, Zhonggui Chen, and Xiaohu Guo. Sliver-suppressing tetrahedral mesh optimization with gradient-based shape matching energy. *Computer Aided Geometric Design*, 52:247–261, 2017.

[PPTSH14] Daniele Panozzo, Enrico Puppo, Marco Tarini, and Olga Sorkine-Hornung. Frame fields: Anisotropic and non-orthogonal cross fields. *ACM Transactions on Graphics*, 33(4):134:1–134:11, 2014.

[RLWB16] Mael Rouxel-Labbé, Mathijs Wintraecken, and J-D Boissonnat. Discretized Riemannian Delaunay triangulations. *Procedia Engineering*, 163:97–109, 2016.

[SA07] Olga Sorkine and Marc Alexa. As-rigid-as-possible surface modeling. In *SGP '07 Proceedings of the fifth Eurographics symposium on Geometry processing*, pages 109–116, 2007.

[SG95] K. Shimada and D. C. Gossard. Bubble mesh: Automated triangular meshing of non-manifold geometry by sphere packing. In *Proceedings of the 3rd ACM Symposium on Solid Modeling and Applications*, pages 409–419, 1995.

[Sim94] R. B. Simpson. Anisotropic mesh transformations and optimal error control. *Applied Numerical Mathematics*, 14(1–3):183–198, 1994.

[SP04] Robert W. Sumner and Jovan Popović. Deformation transfer for triangle meshes. *ACM Transactions on Graphics*, 23(3):399–405, 2004.

[SRM14] E. Sauvage, J.F. Remacle, and E. Marchandise. Metric field construction for anisotropic mesh adaptation with application to blood flow simulations. *International Journal for Numerical Methods in Biomedical Engineering*, 30(11):1326–1346, 2014.

[SYI97] K. Shimada, A. Yamada, and T. Itoh. Anisotropic triangular meshing of parametric surfaces via close packing of ellipsoidal bubbles. In *6th International Meshing Roundtable*, pages 375–390, 1997.

[TSA09] Jane Tournois, Rahul Srinivasan, and Pierre Alliez. Perturbing slivers in 3D Delaunay meshes. *Proceedings of the 18th international meshing roundtable*, pages 157–173, 2009.

[VCP08] S. Valette, J. M. Chassery, and R. Prost. Generic remeshing of 3D triangular meshes with metric-dependent discrete Voronoi diagrams. *IEEE Transactions on Visualization and Computer Graphics*, 14(2):369–381, 2008.

[Ver12] Nakul Verma. Distance preserving embeddings for general n-dimensional manifolds. *Journal of Machine Learning Research volume*, 14:2415–2448, 2012.

[YLL+09] D. Yan, B. Lévy, Y. Liu, F. Sun, and W. Wang. Isotropic remeshing with fast and exact computation of restricted Voronoi diagram. *Computer Graphics Forum*, 28(5):1445–1454, 2009.

[YS00] S. Yamakawa and K. Shimada. High quality anisotropic tetrahedral mesh generation via packing ellipsoidal bubbles. In *9th International Meshing Roundtable*, pages 263–273, 2000.

[YWLL13] Dong-Ming Yan, Wenping Wang, Bruno Lévy, and Yang Liu. Efficient computation of clipped Voronoi diagram for mesh generation. *Computer-Aided Design*, 45(4):843–852, 2013.

[ZGW+13] Zichun Zhong, Xiaohu Guo, Wenping Wang, Bruno Lévy, Feng Sun, Yang Liu, and Weihua Mao. Particle-based anisotropic surface meshing. *ACM Trans. Graph.*, 32(4):99:1–99:14, 2013.

[ZSJG14] Zichun Zhong, Liang Shuai, Miao Jin, and Xiaohu Guo. Anisotropic surface meshing with conformal embedding. *Graphical Models*, 76(5):468–483, 2014.